Statistical Relational learning and Probabilistic Programming

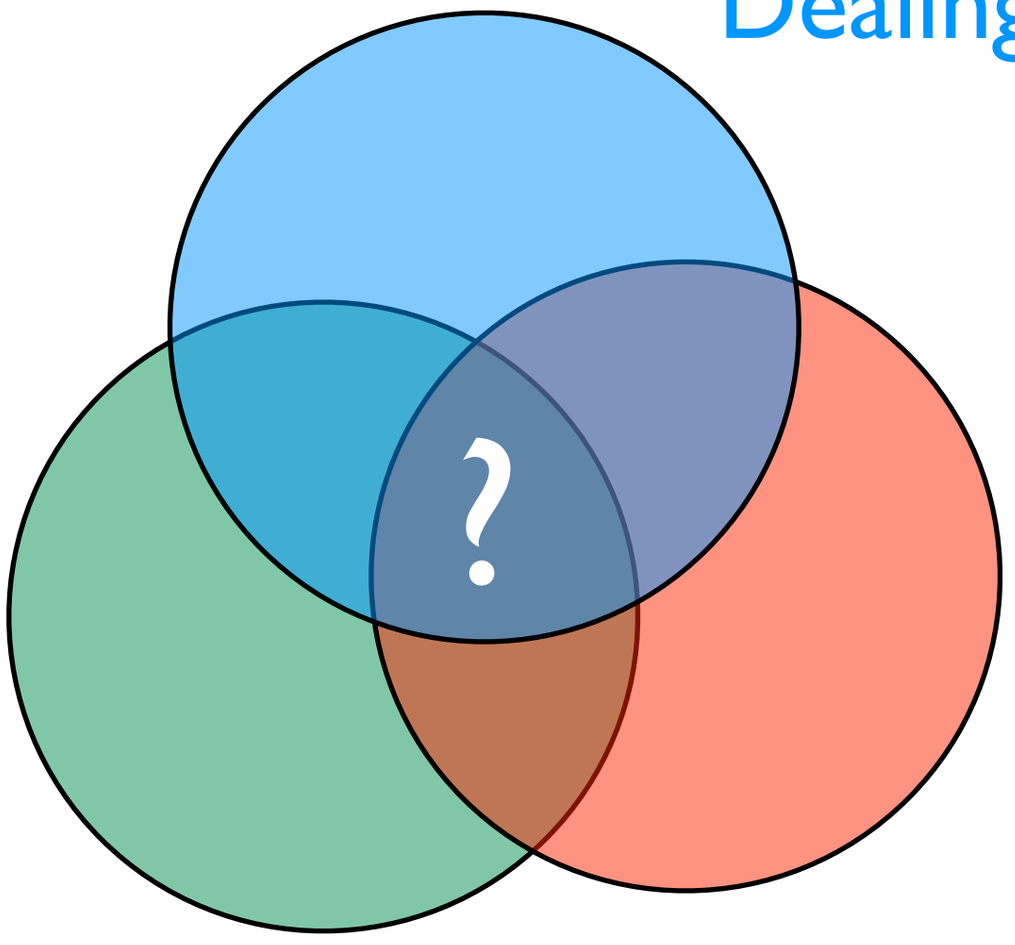
Luc De Raedt, Anton Dries, and Angelika Kimmig
Also many slides from Guy Van den Brouck

https://dtai.cs.kuleuven.be/problog/wasp17-tutorial.html

A key question in AI:

Dealing with uncertainty

Reasoning with relational data



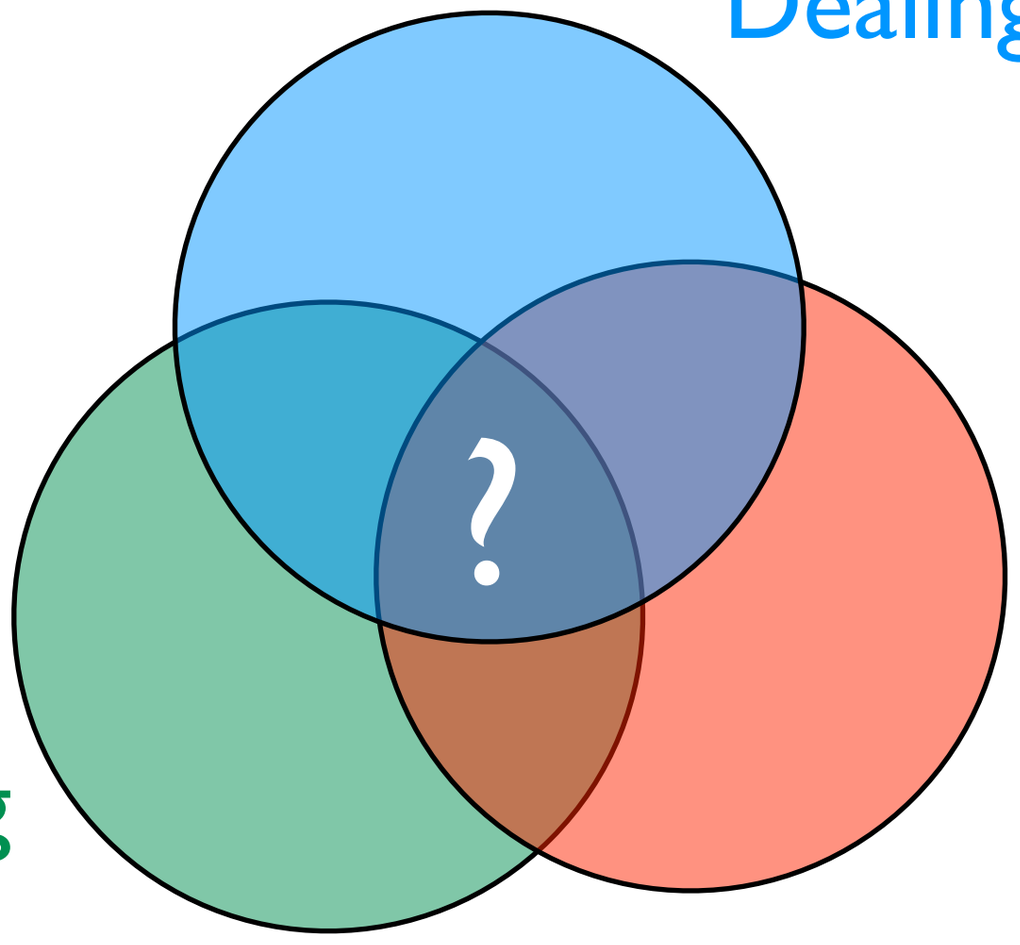
Learning

A key question in AI:

Dealing with uncertainty

Reasoning with relational data

- logic
- databases
- programming
- ...

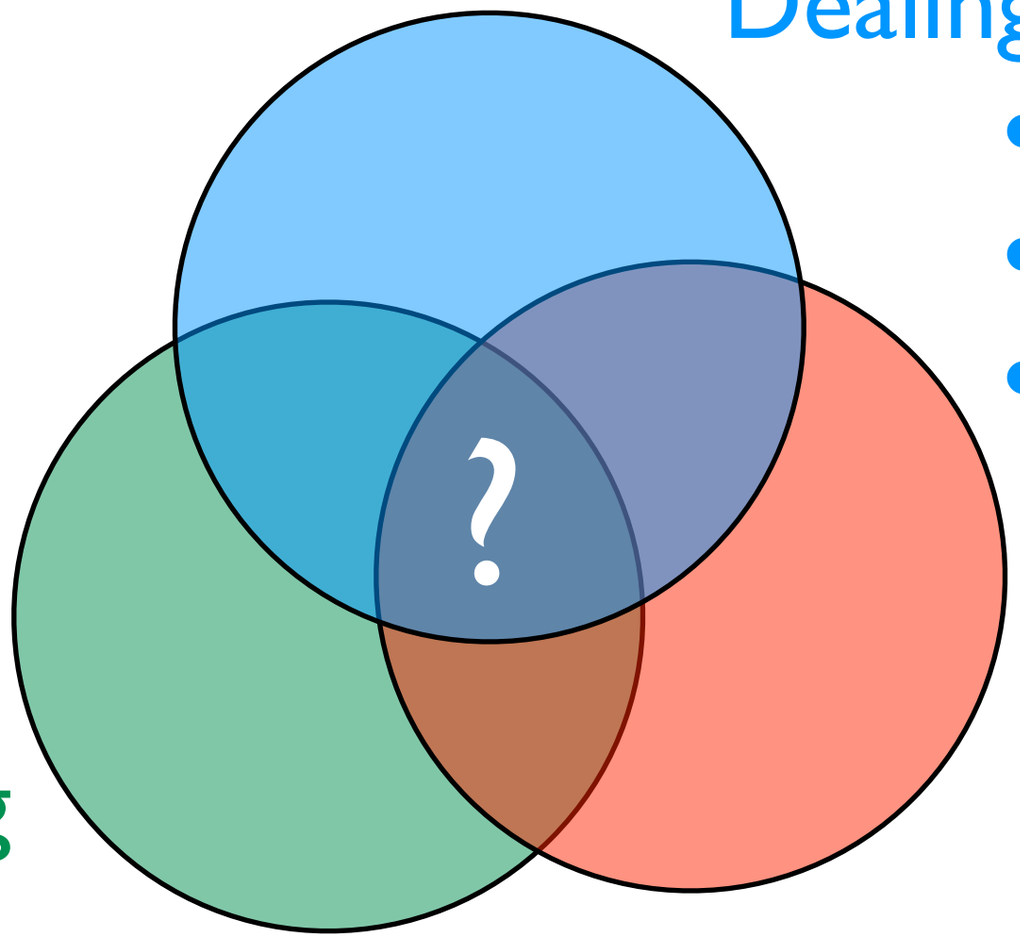


Learning

A key question in AI:

Reasoning with relational data

- logic
- databases
- programming
- ...



Dealing with uncertainty

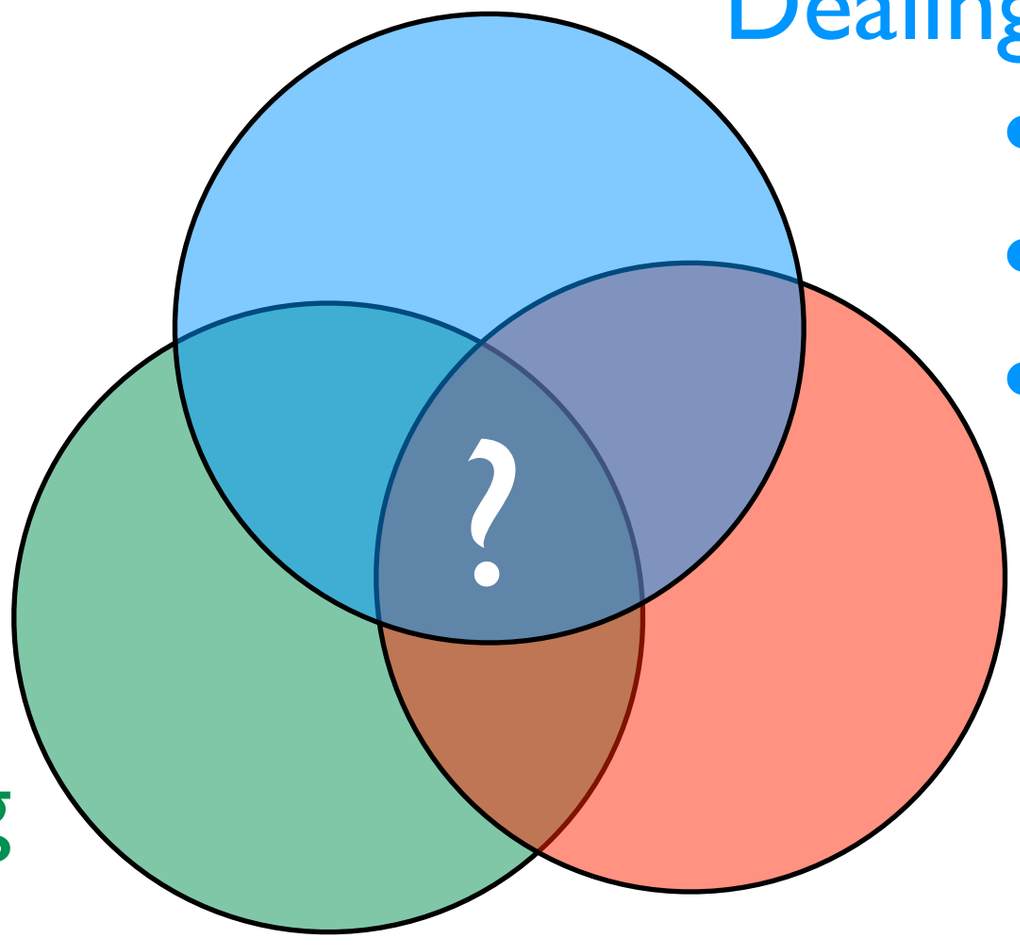
- probability theory
- graphical models
- ...

Learning

A key question in AI:

Reasoning with relational data

- logic
- databases
- programming
- ...



Dealing with uncertainty

- probability theory
- graphical models
- ...

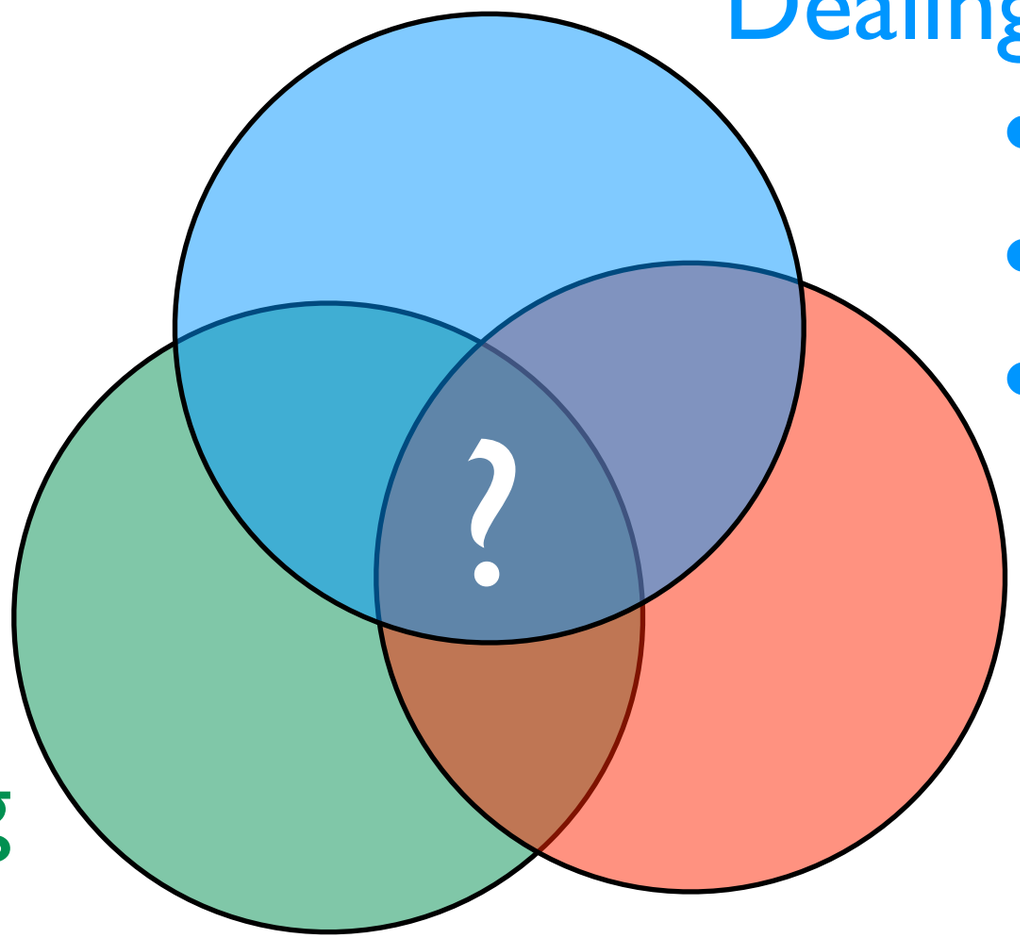
Learning

- parameters
- structure

A key question in AI:

Reasoning with relational data

- logic
- databases
- programming
- ...



Dealing with uncertainty

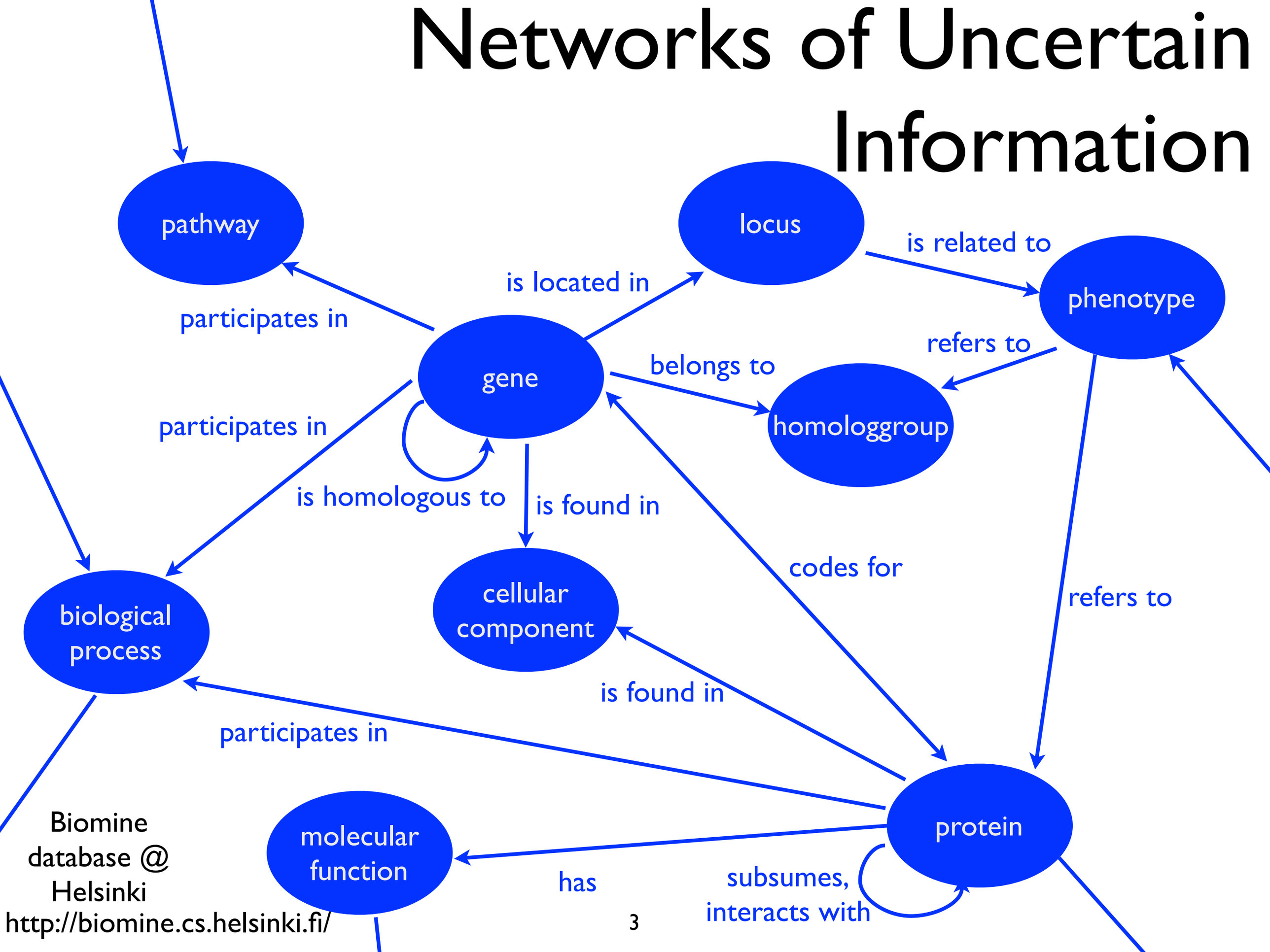
- probability theory
- graphical models
- ...

Learning

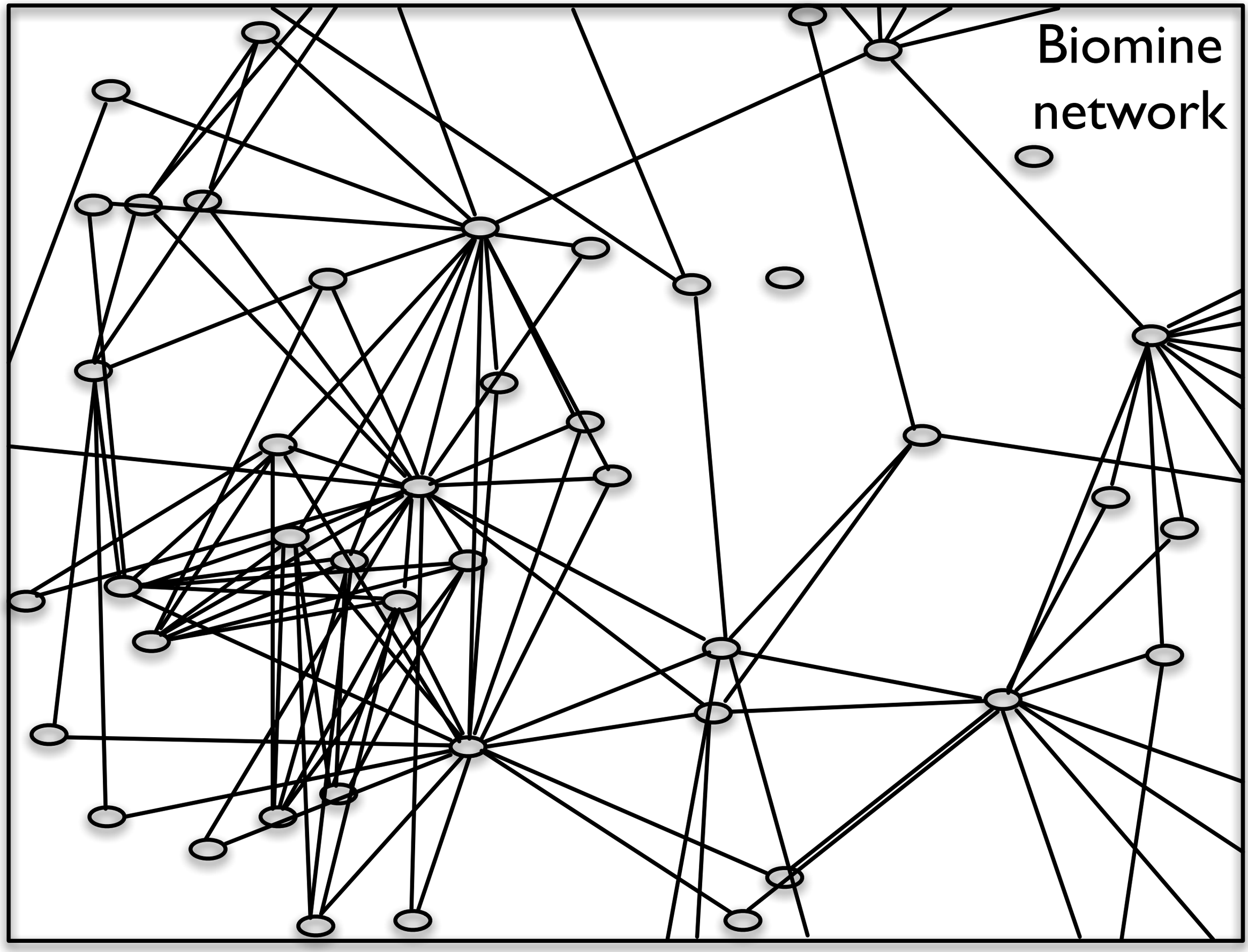
- parameters
- structure

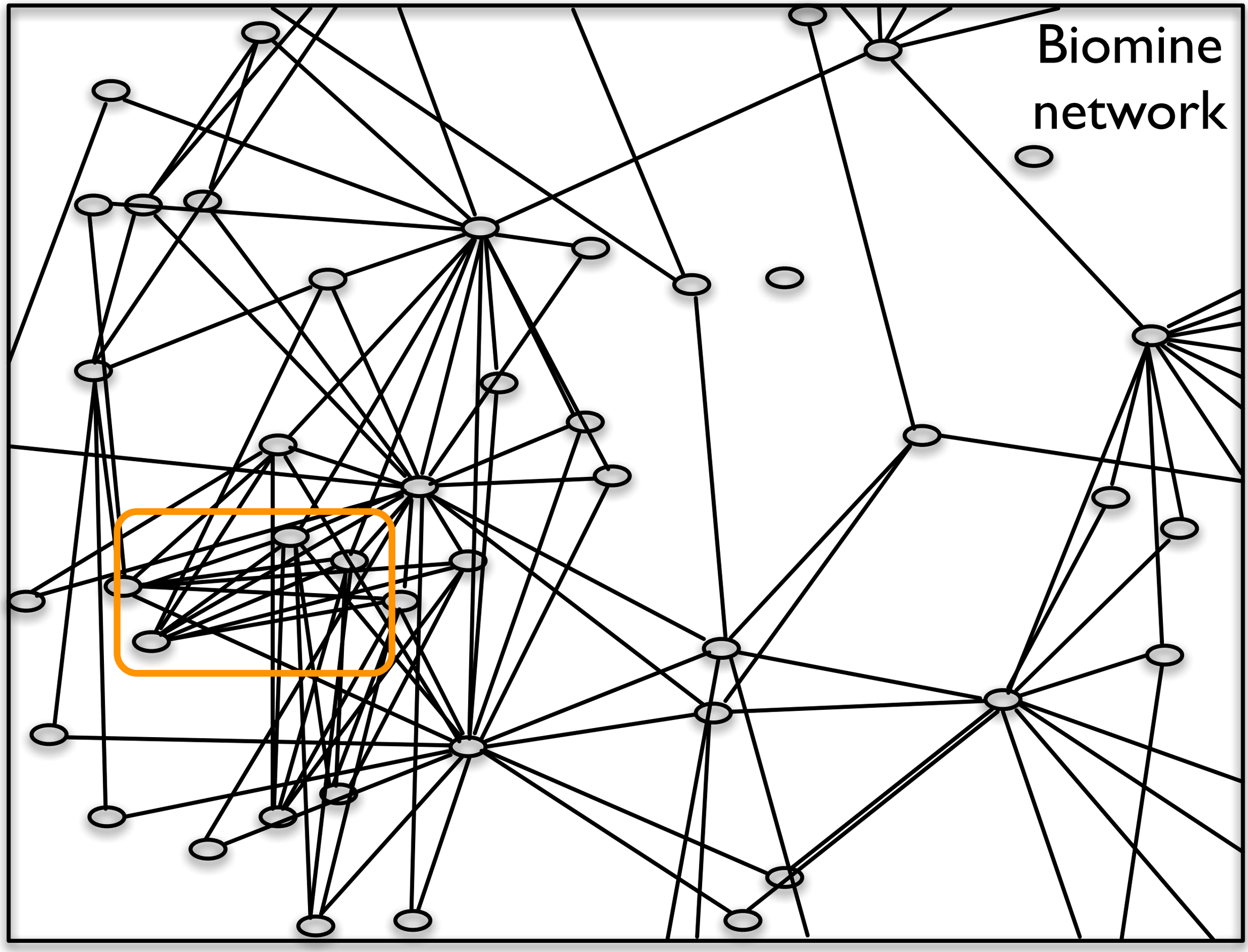
Statistical relational learning, probabilistic logic learning, probabilistic programming, ...

Networks of Uncertain Information

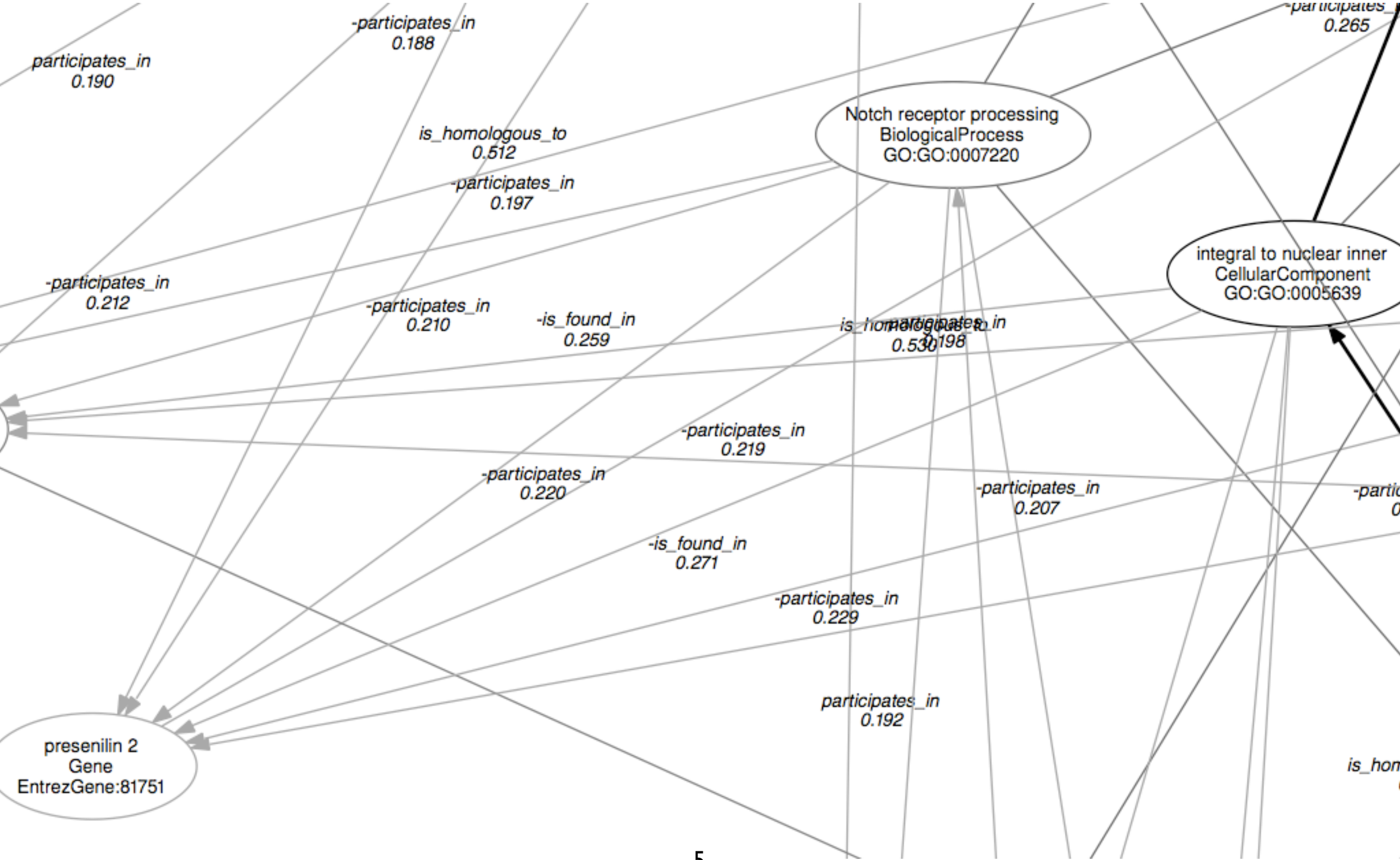


Biomine database @ Helsinki
http://biomine.cs.helsinki.fi/


Biomine
network

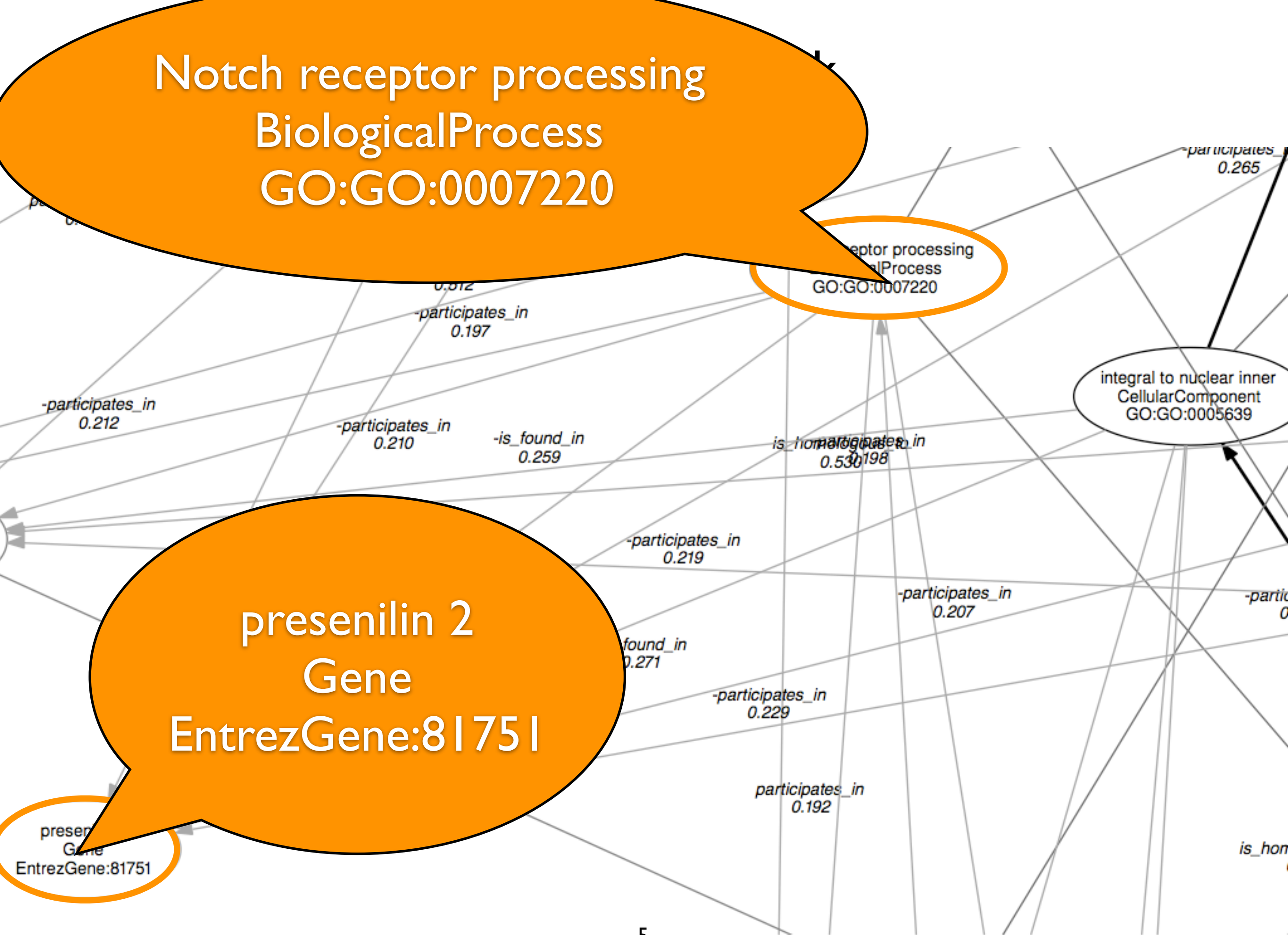

Biomine
network

Biomine Network

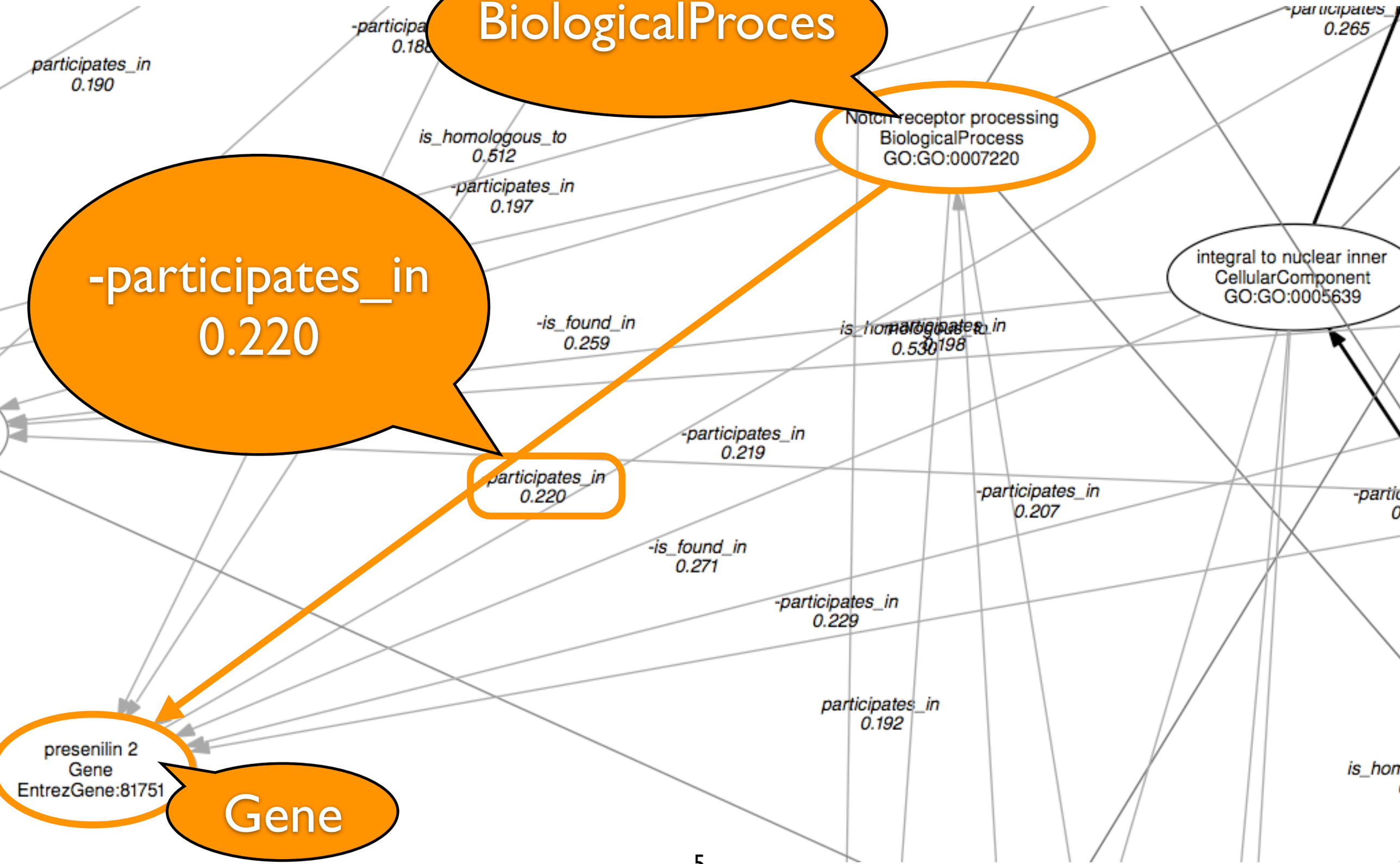


Notch receptor processing
BiologicalProcess
GO:GO:0007220

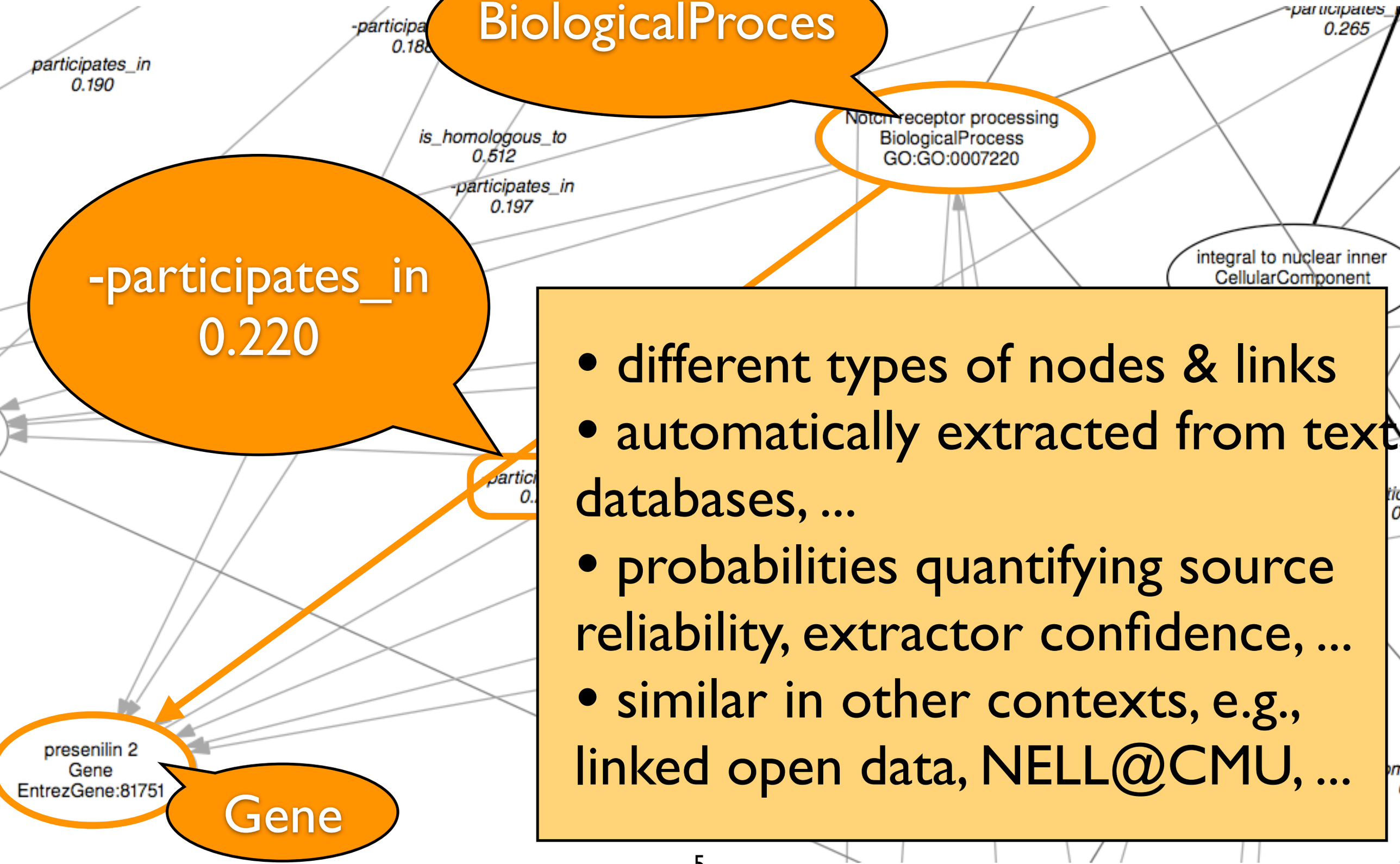
presenilin 2
Gene
EntrezGene:81751



Biomine Network



Biomine Network



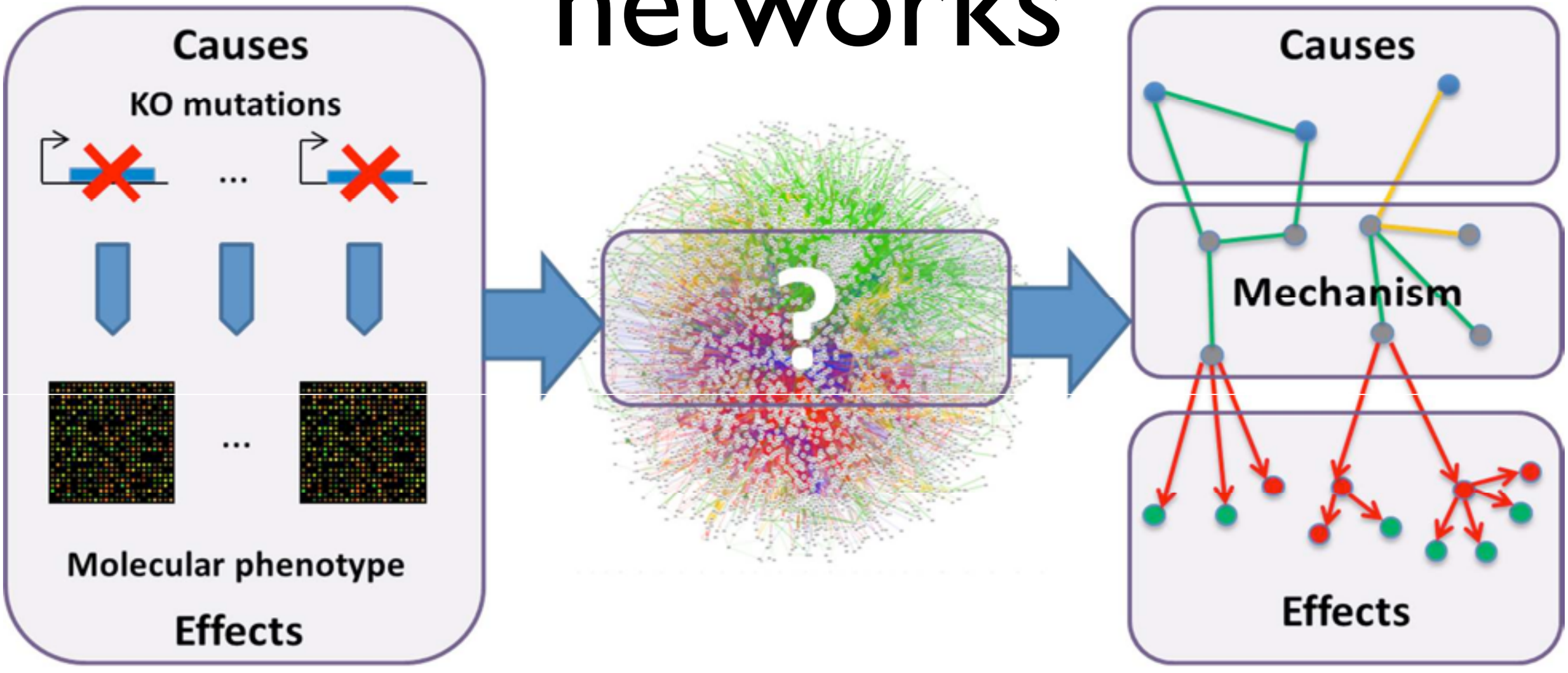
BiologicalProces

-participates_in
0.220

Gene

- different types of nodes & links
- automatically extracted from text databases, ...
- probabilities quantifying source reliability, extractor confidence, ...
- similar in other contexts, e.g., linked open data, NELL@CMU, ...

Molecular interaction networks



Can we find the mechanism connecting causes to effects?

[De Maeyer et al, Molecular Biosystems 13]

Example: Information Extraction

Recently-Learned Facts twitter

Refresh

instance	iteration	date learned	confidence
kelly_andrews is a female	826	29-mar-2014	98.7
investment_next_year is an economic sector	829	10-apr-2014	95.3
shibenik is a geopolitical entity that is an organization	829	10-apr-2014	97.2
quality_web_design_work is a character trait	826	29-mar-2014	91.0
mercedes_benz_cls_by_carlsson is an automobile manufacturer	829	10-apr-2014	95.2
social_work is an academic program at the university rutgers_university	827	02-apr-2014	93.8
dante wrote the book the_divine_comedy	826	29-mar-2014	93.8
willie_aames was born in the city los_angeles	831	16-apr-2014	100.0
kitt_peak is a mountain in the state or province arizona	831	16-apr-2014	96.9
greenwich is a park in the city london	831	16-apr-2014	100.0

NELL: http://rtw.ml.cmu.edu/rtw/

Example: Information Extraction

Recently-Learned Facts twitter Refresh

instance	iteration	date learned	confidence
kelly_andrews is a female	826	29-mar-2014	98.7
investment_next_year is an economic sector	829	10-apr-2014	95.3
shibenik is a geopolitical entity that is an organization	829	10-apr-2014	97.2
quality_web_design_work is a character trait	826	29-mar-2014	91.0
mercedes_benz_cls_by_carlsson is an automobile manufacturer	829	10-apr-2014	95.2
social_work is an academic program at the university rutgers_university	827	02-apr-2014	93.8
dante wrote the book the_divine_comedy	826	29-mar-2014	93.8
willie_aames was born in the city los_angeles	831	16-apr-2014	100.0
kitt_peak is a mountain in the state or province arizona	831	16-apr-2014	96.9
greenwich is a park in the city london	831	16-apr-2014	100.0

instances for many different relations

NELL: http://rtw.ml.cmu.edu/rtw/

Example: Information Extraction

Recently-Learned Facts twitter Refresh

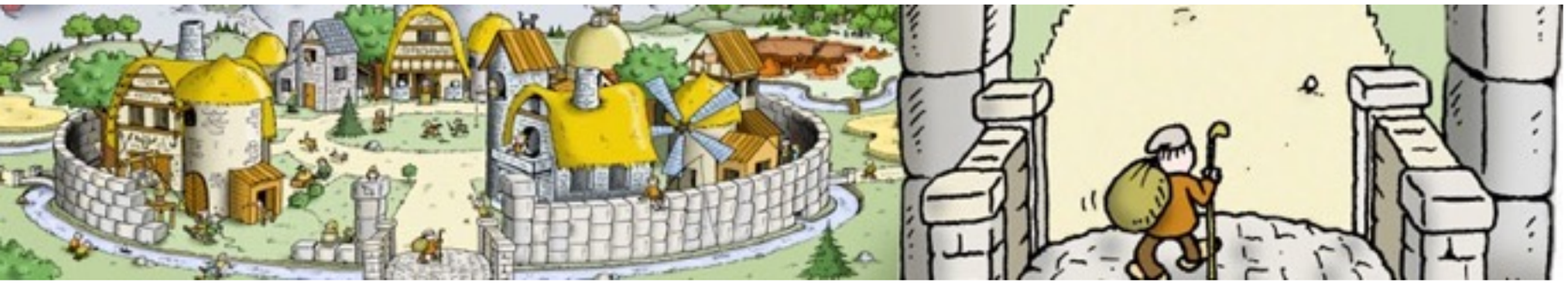
instance	iteration	date learned	confidence
kelly_andrews is a female	826	29-mar-2014	98.7
investment_next_year is an economic sector	829	10-apr-2014	95.3
shibenik is a geopolitical entity that is an organization	829	10-apr-2014	97.2
quality_web_design_work is a character trait	826	29-mar-2014	91.0
mercedes_benz_cls_by_carlsson is an automobile manufacturer	829	10-apr-2014	95.2
social_work is an academic program at the university rutgers_university	827	02-apr-2014	93.8
dante wrote the book the_divine_comedy	826	29-mar-2014	93.8
willie_aames was born in the city los_angeles	831	16-apr-2014	100.0
kitt_peak is a mountain in the state or province arizona	831	16-apr-2014	96.9
greenwich is a park in the city london	831	16-apr-2014	100.0

instances for many different relations

degree of certainty

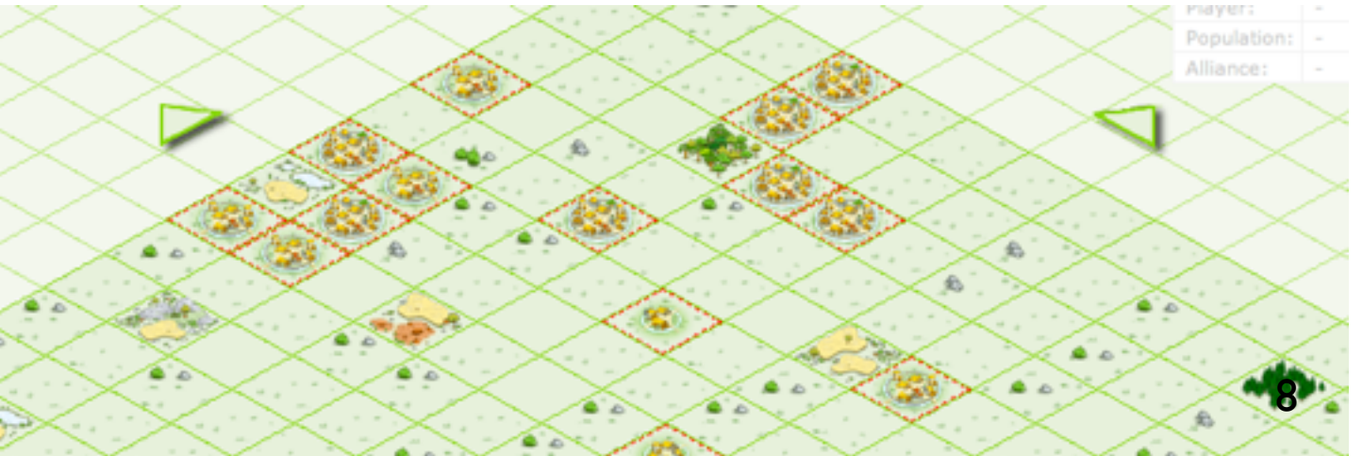
NELL: http://rtw.ml.cmu.edu/rtw/

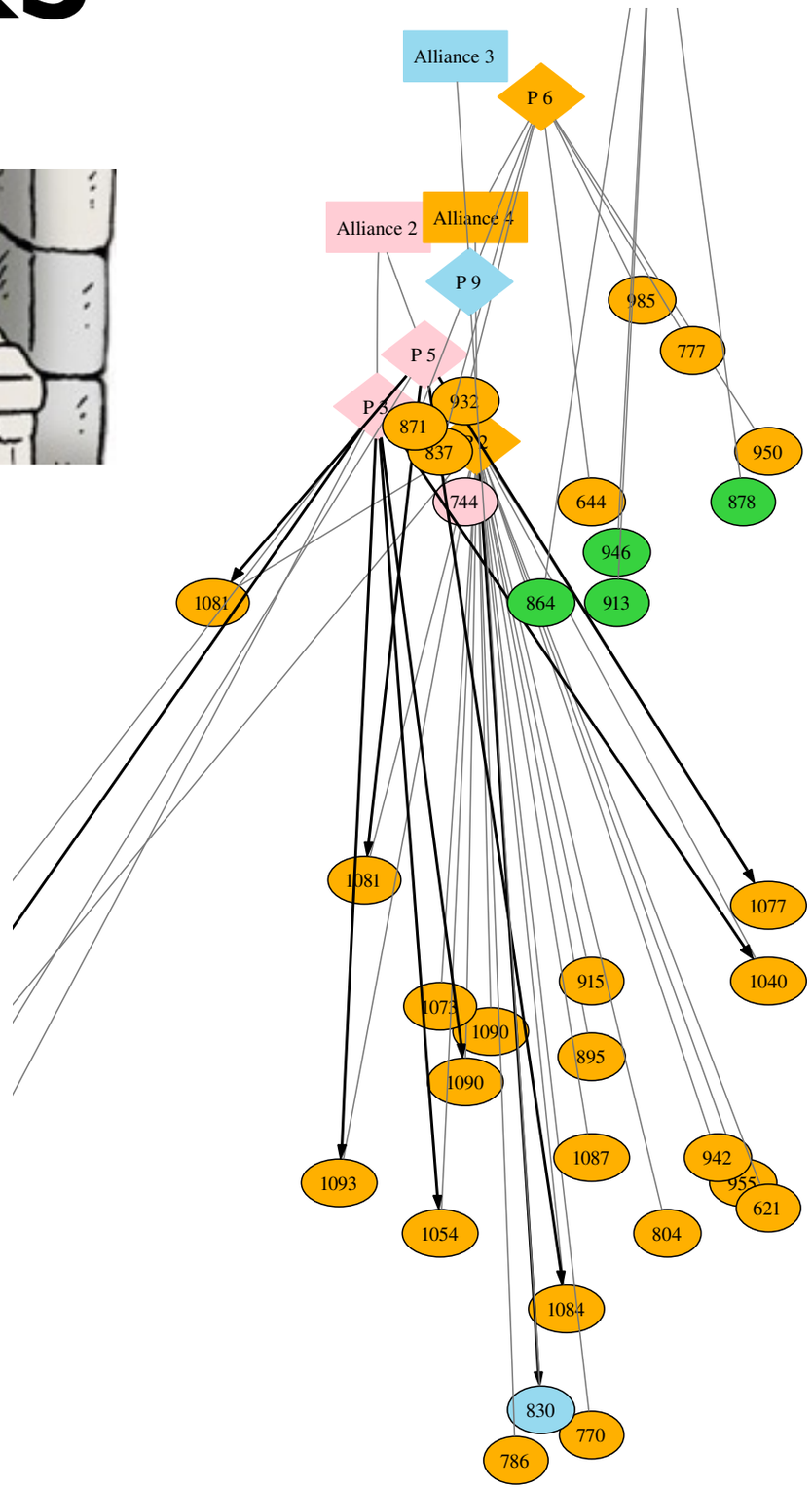
Dynamic networks



Travian: A massively multiplayer real-time strategy game

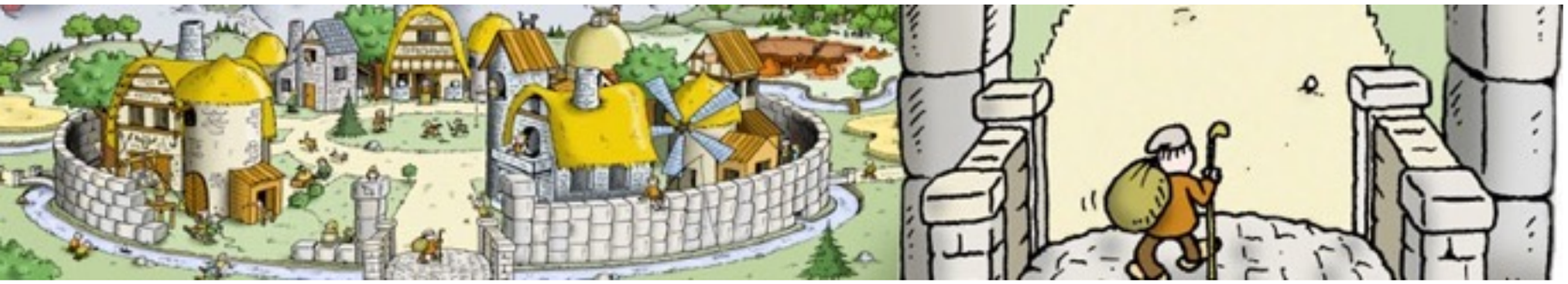
Can we build a model
of this world ?
Can we use it for playing
better ?





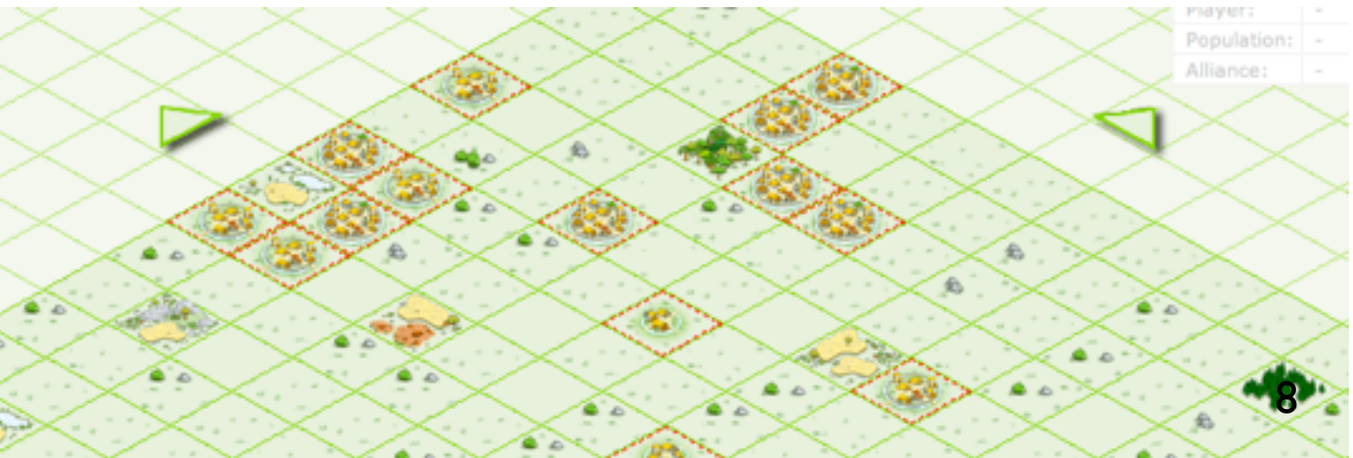
[Thon et al, MLJ 11]

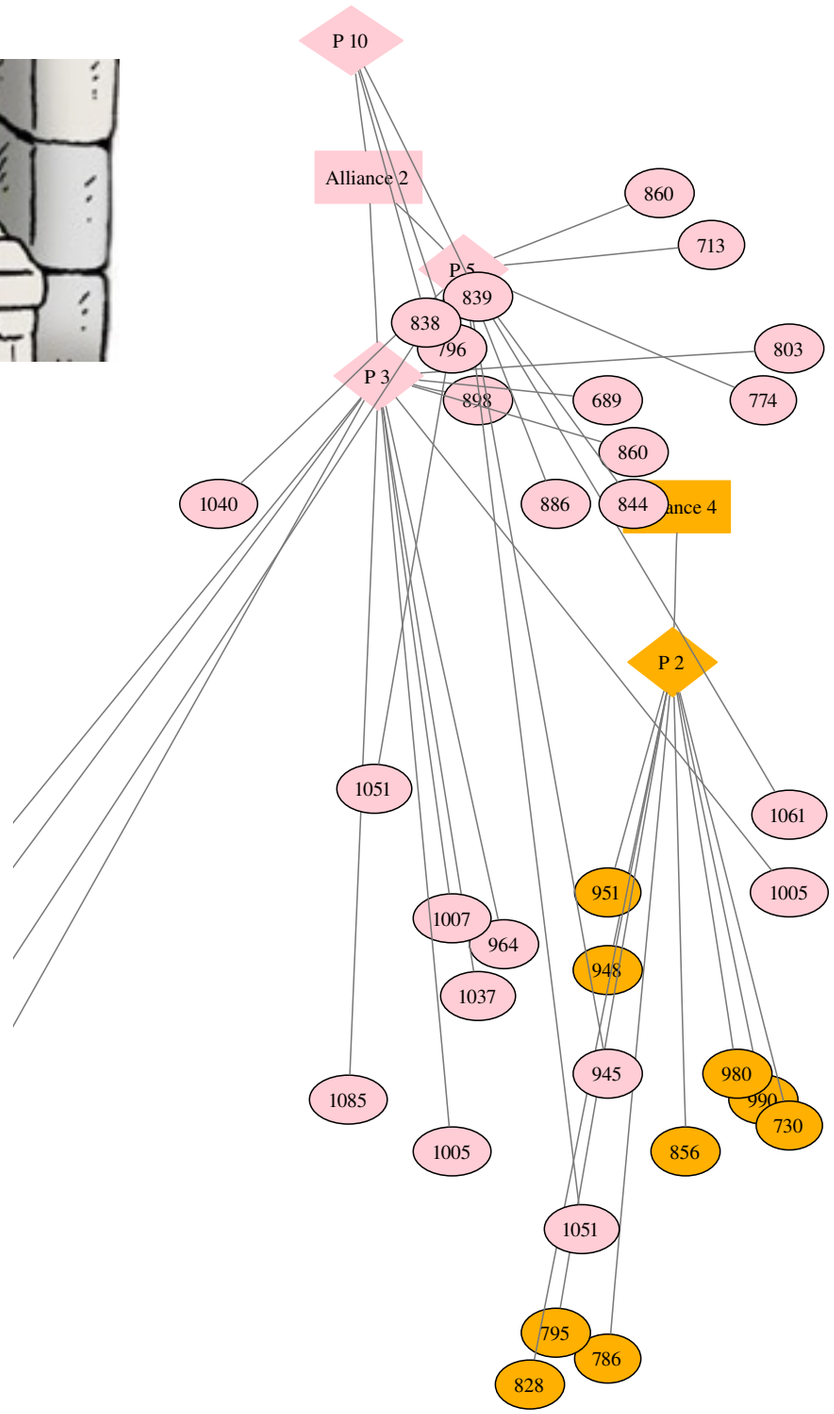
Dynamic networks



Travian: A massively multiplayer real-time strategy game

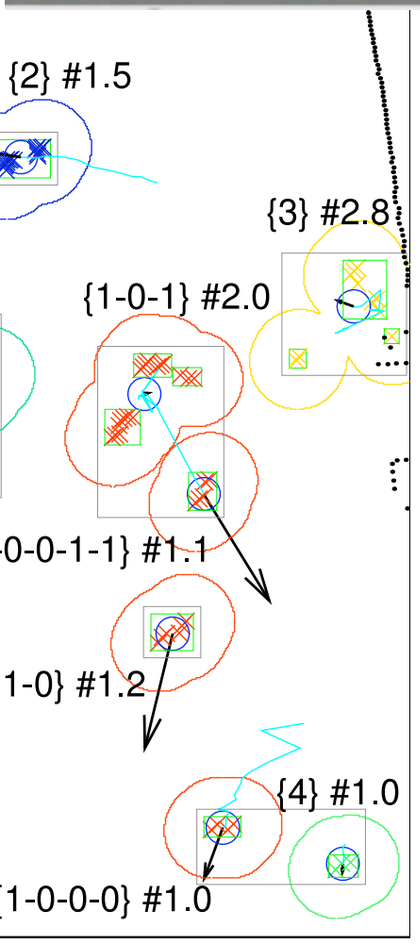
Can we build a model
of this world ?
Can we use it for playing
better ?



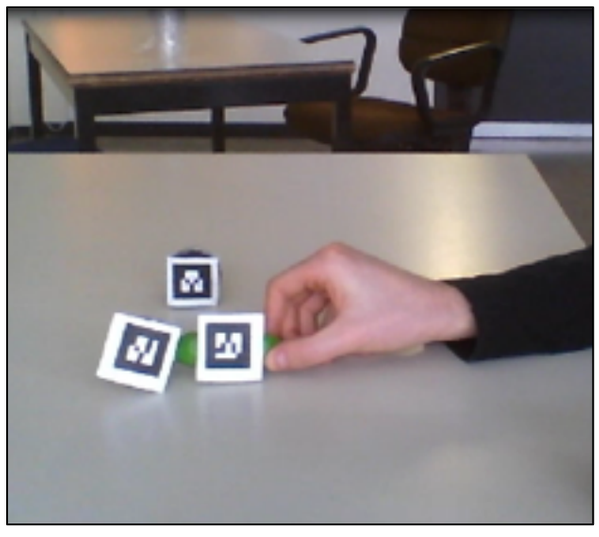


[Thon et al, MLJ 11]

Analyzing Video Data



- Track people or objects over time? Even if temporarily hidden?
- Recognize activities?
- Infer object properties?



[Skarlatidis et al, TPLP 14;
Nitti et al, IROS 13, ICRA 14]

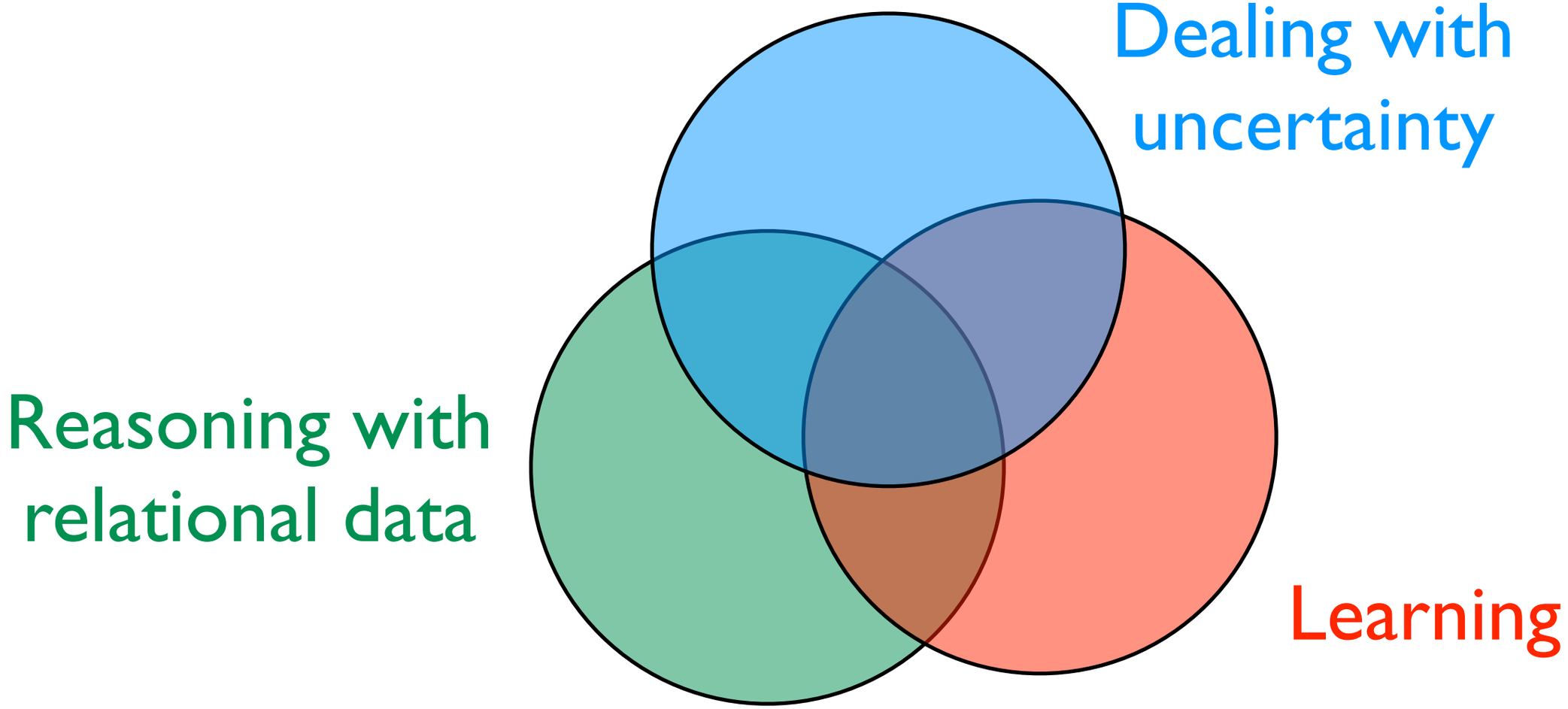
Answering Probability Questions

Mike has a bag of marbles with 4 white, 8 blue, and 6 red marbles. He pulls out one marble from the bag and it is red. What is the probability that the second marble he pulls out of the bag is white?

The answer is 0.235941.

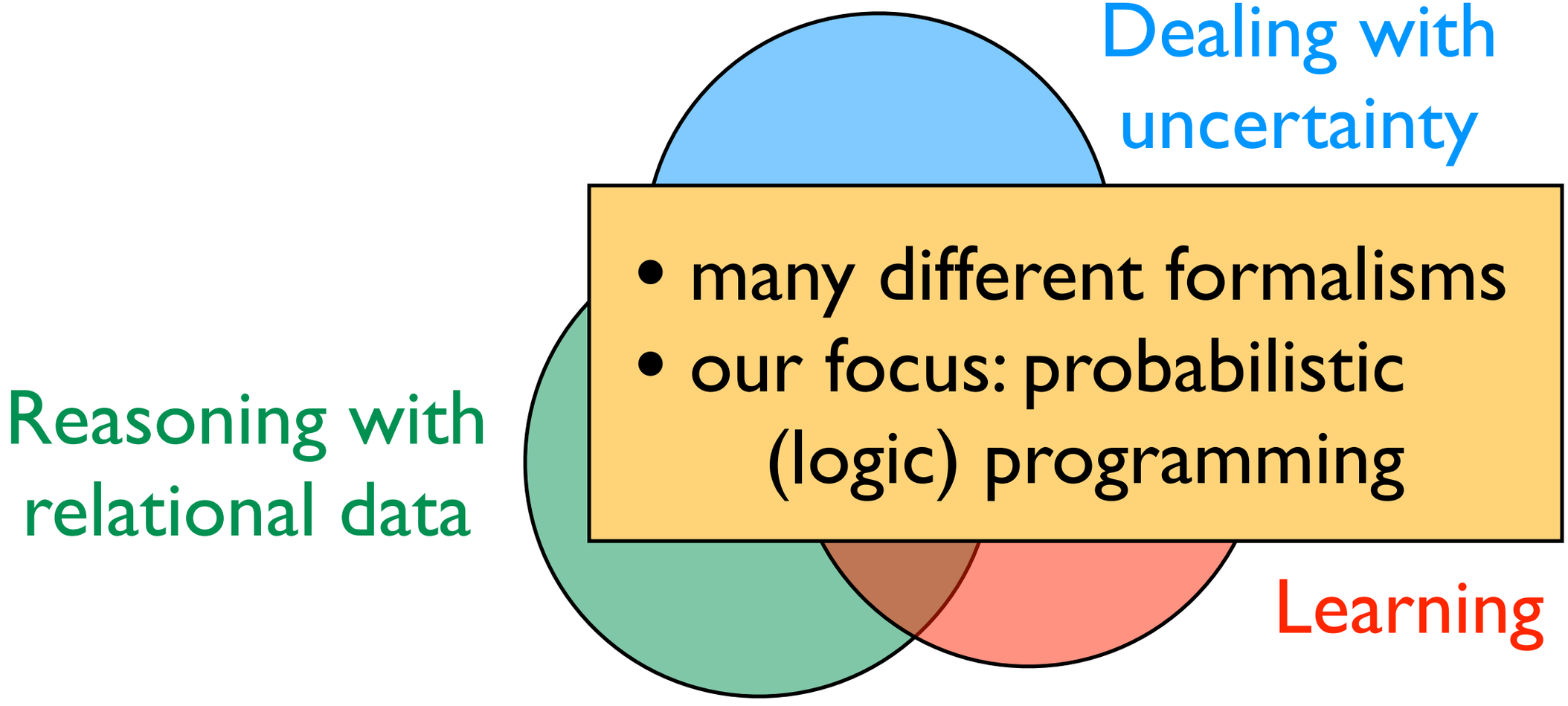
[Dries et al., IJCAI 17]

Common theme



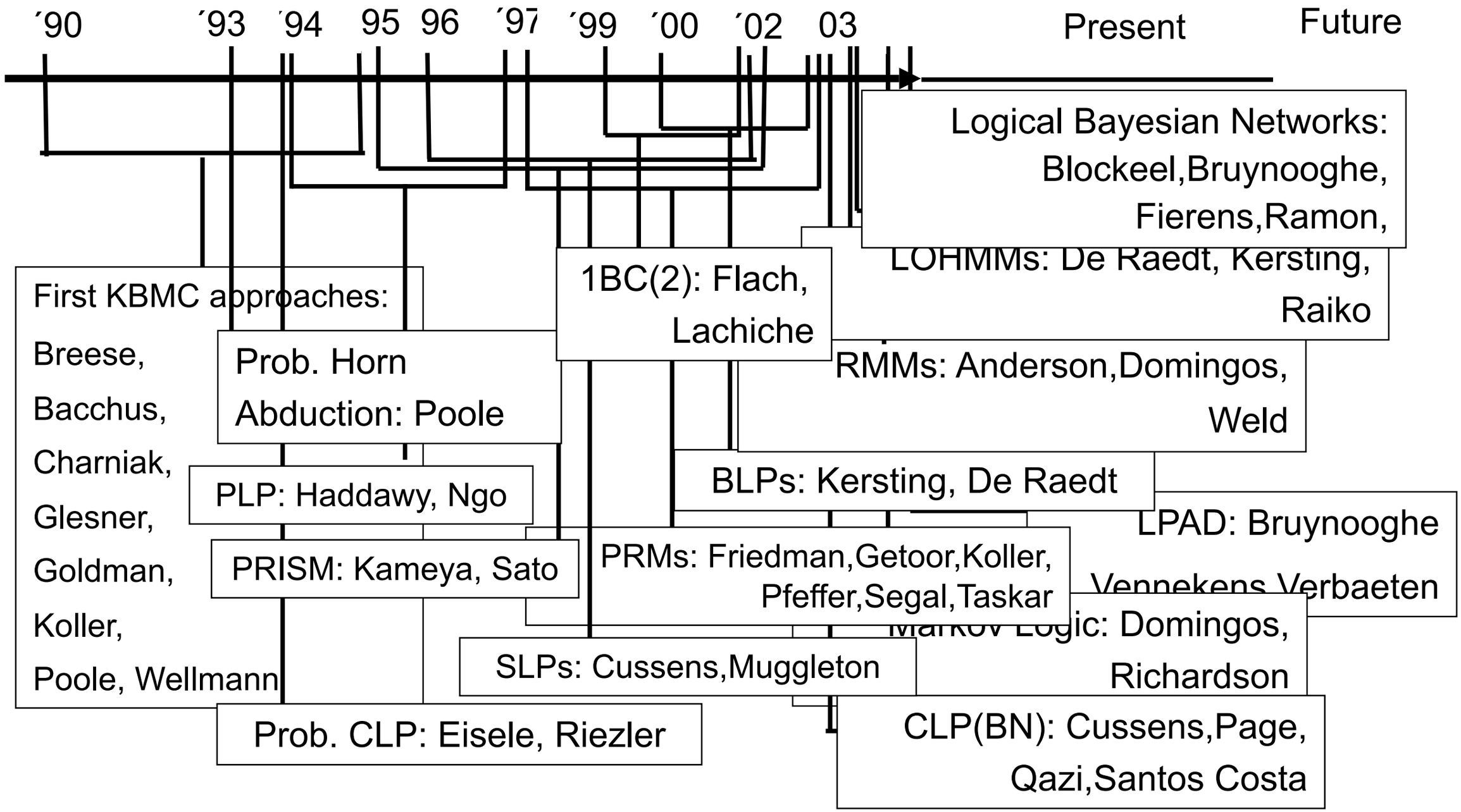
Statistical relational learning, probabilistic logic learning, probabilistic programming, ...

Common theme



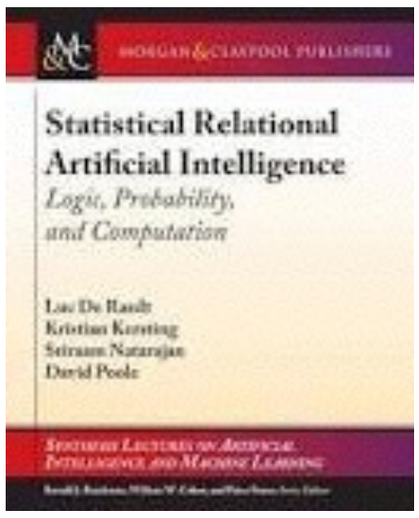
Statistical relational learning, probabilistic logic learning, probabilistic programming, ...

Some formalisms



Many different angles

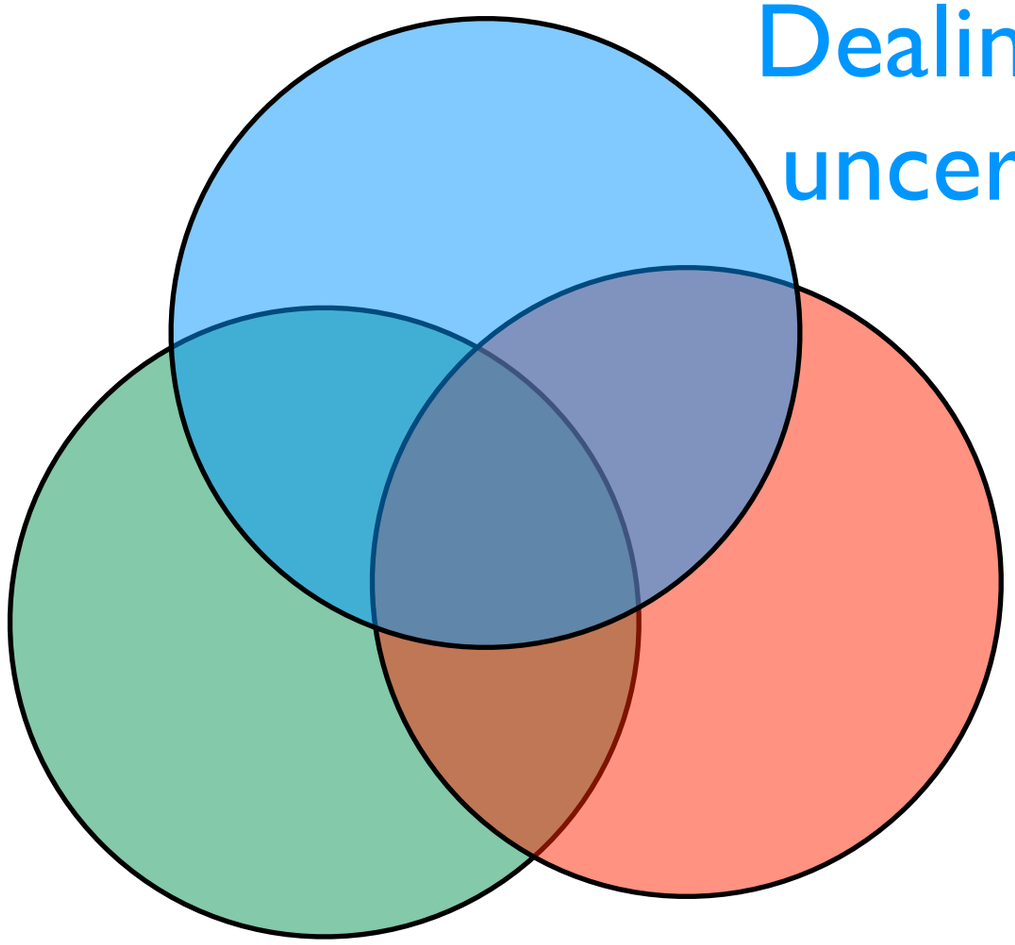
- Probabilistic programming
  - Logic programming and probabilistic databases (ProbLog and DS as representatives)
  - Functional and imperative (Church as representatives)
- Statistical relational AI and learning
  - Markov Logic
  - Relational Bayesian Networks (and variants)



ProbLog

probabilistic Prolog

Dealing with uncertainty



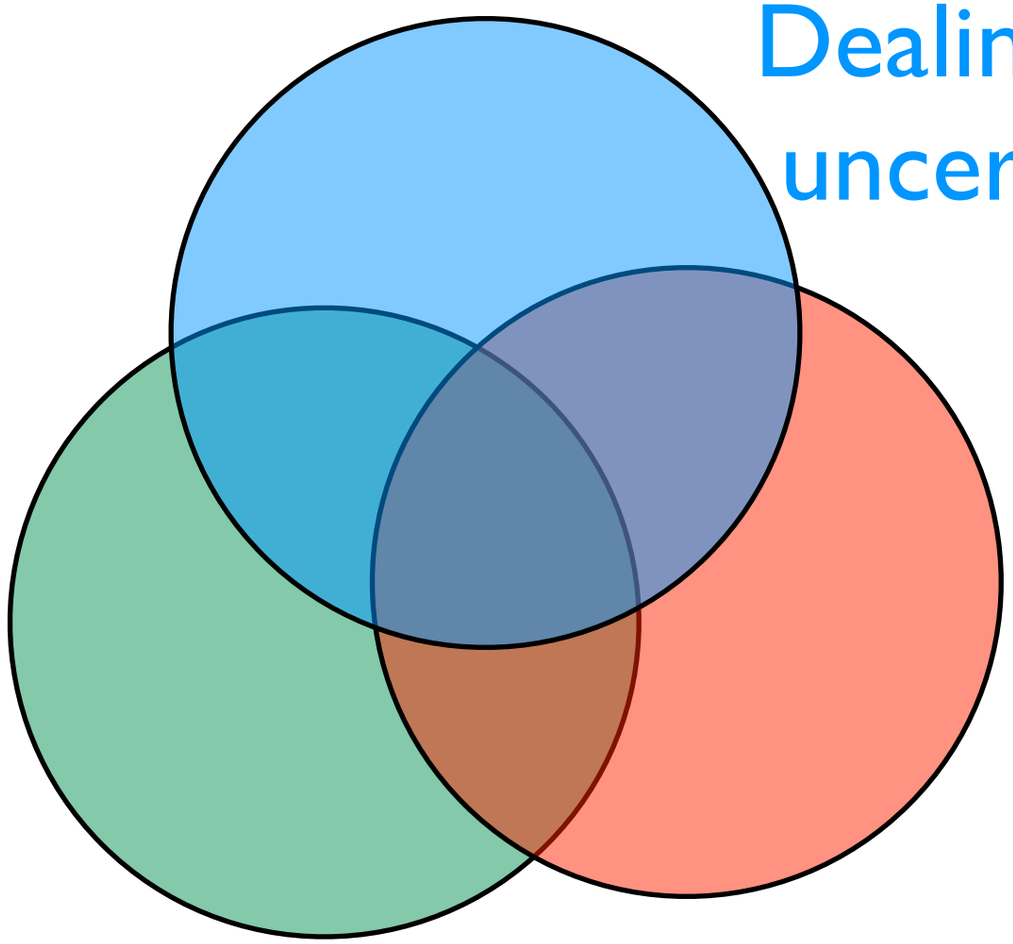
Reasoning with relational data

Learning

http://dtai.cs.kuleuven.be/problog/

ProbLog

probabilistic Prolog

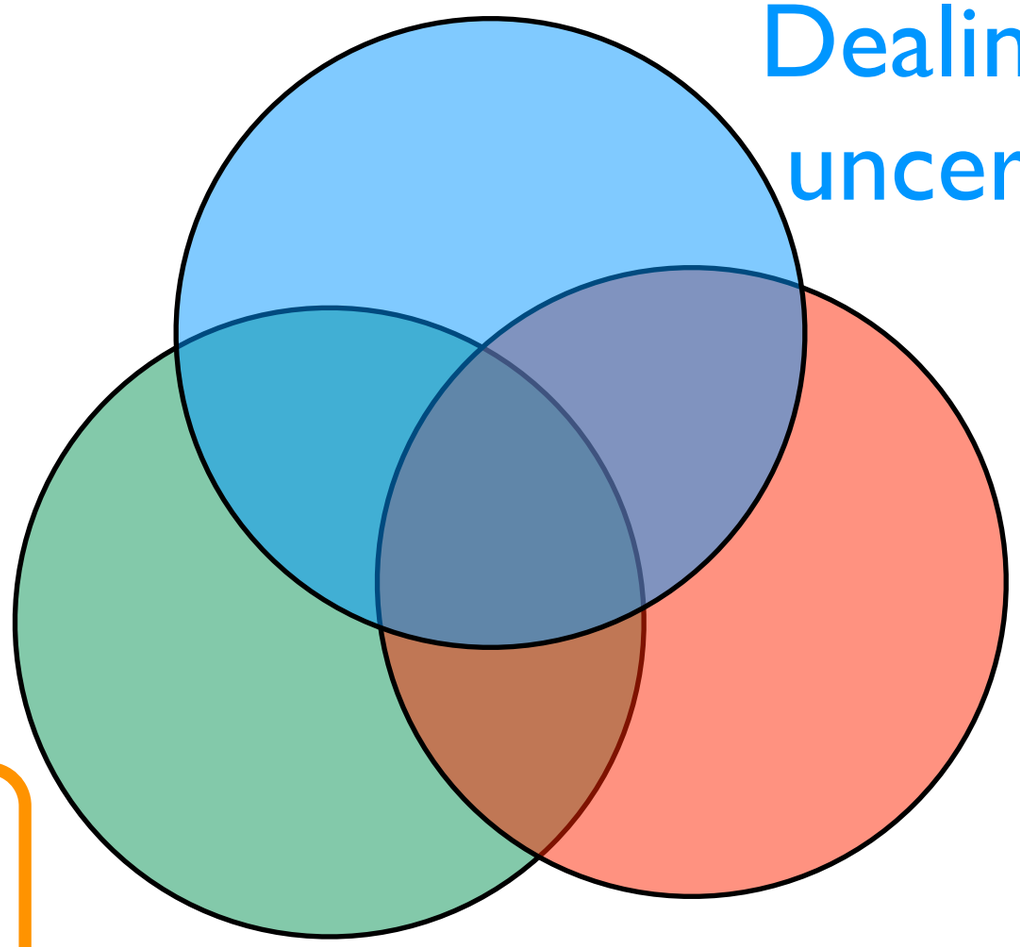



```
stress(ann).
influences(ann,bob).
influences(bob,carl).

smokes(X) :- stress(X).
smokes(X) :-
     influences(Y,X), smokes(Y).
```


http://dtai.cs.kuleuven.be/problog/

ProbLog

probabilistic Prolog




```
stress(ann).
influences(ann,bob).
influences(bob,carl).
```

one world

```
smokes(X) :- stress(X).
smokes(X) :-
     influences(Y,X), smokes(Y).
```

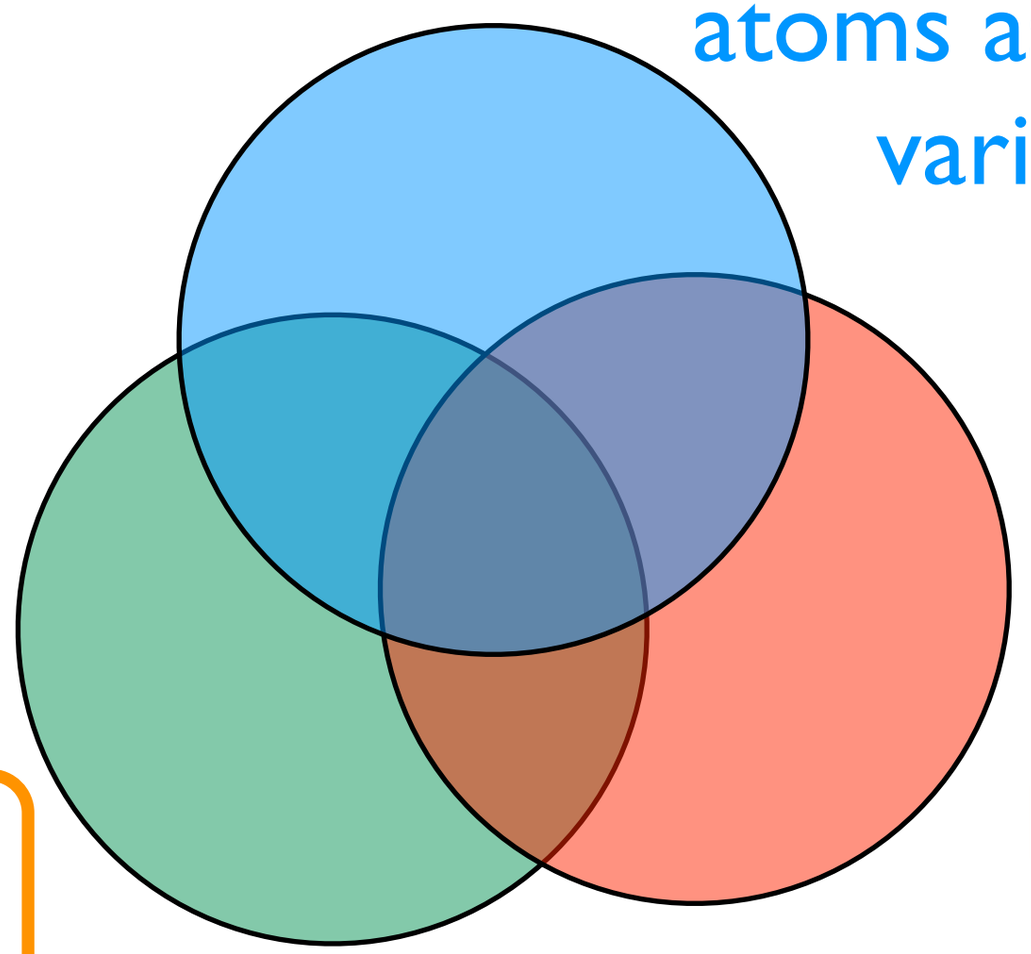

http://dtai.cs.kuleuven.be/problog/

ProbLog

probabilistic Prolog

```
0.8::stress(ann).
0.6::influences(ann,bob).
0.2::influences(bob,carl).
```

atoms as random variables



Prolog / logic programming

Learning

```
stress(ann).
influences(ann,bob).
influences(bob,carl).
```

one world

```
smokes(X) :- stress(X).
smokes(X) :-
     influences(Y,X), smokes(Y).
```


http://dtai.cs.kuleuven.be/problog/

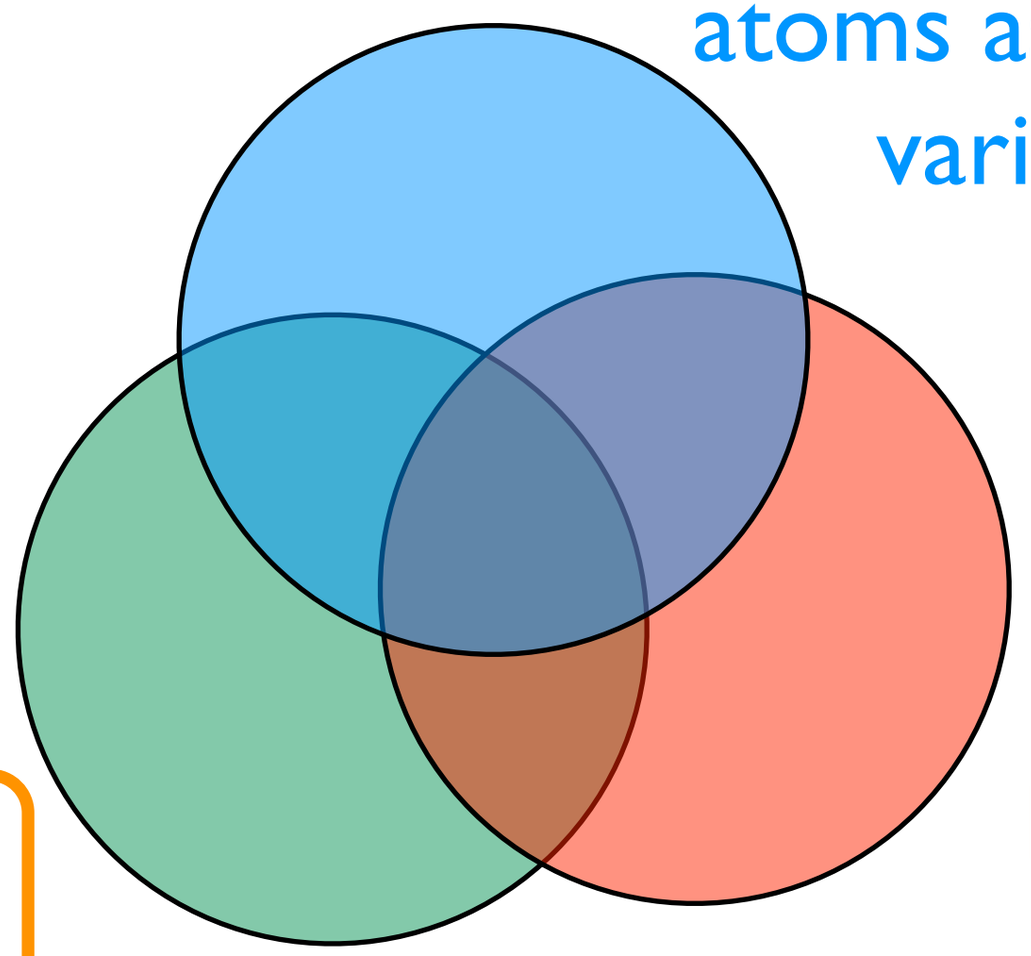
ProbLog

probabilistic Prolog

several possible worlds

```
0.8::stress(ann).
0.6::influences(ann,bob).
0.2::influences(bob,carl).
```

atoms as random variables



Prolog / logic programming

Learning

```
stress(ann).
influences(ann,bob).
influences(bob,carl).
```

one world

```
smokes(X) :- stress(X).
smokes(X) :-
     influences(Y,X), smokes(Y).
```


http://dtai.cs.kuleuven.be/problog/

ProbLog

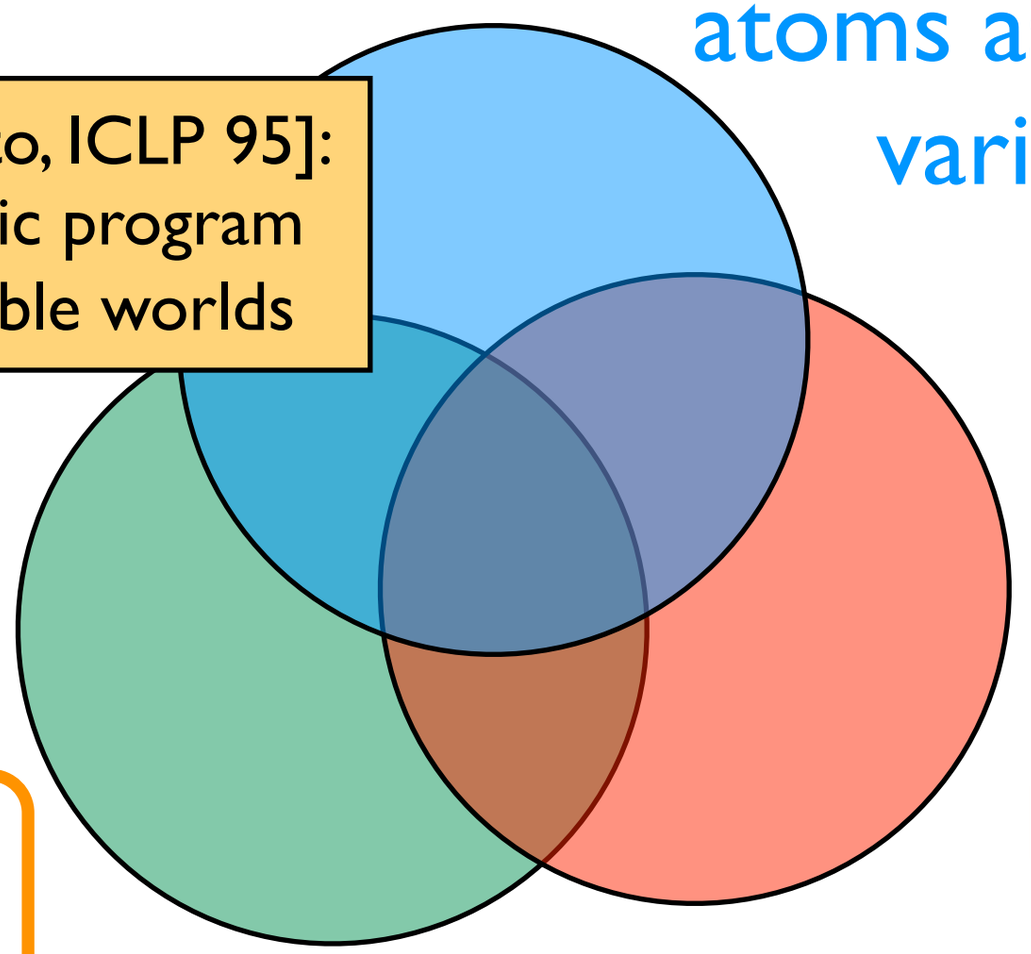
probabilistic Prolog

several possible worlds

```
0.8::stress(ann).
0.6::influences(ann,bob).
0.2::influences(bob,carl).
```

atoms as random variables

Distribution Semantics [Sato, ICLP 95]:
probabilistic choices + logic program
→ distribution over possible worlds



Prolog / logic programming

Learning

```
stress(ann).
influences(ann,bob).
influences(bob,carl).
```

one world

```
smokes(X) :- stress(X).
smokes(X) :-
     influences(Y,X), smokes(Y).
```


http://dtai.cs.kuleuven.be/problog/

ProbLog

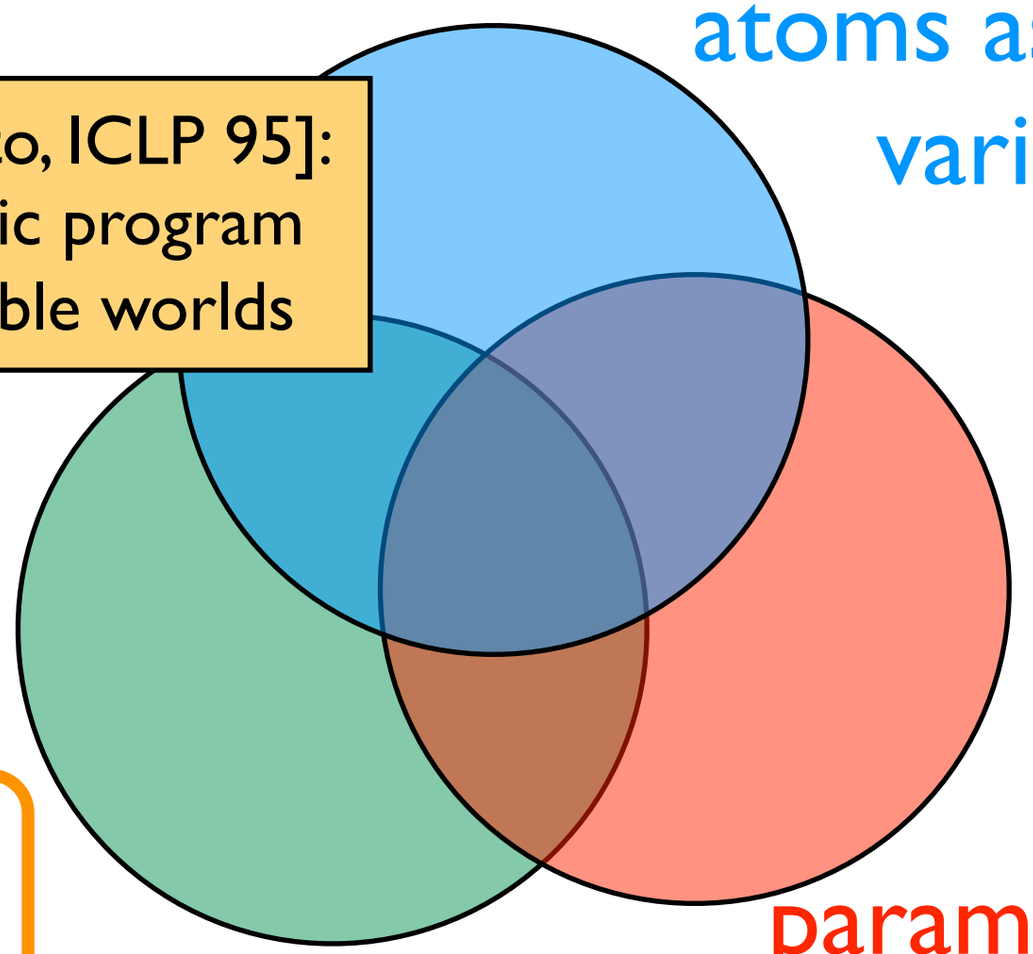
probabilistic Prolog

Distribution Semantics [Sato, ICLP 95]:
probabilistic choices + logic program
→ distribution over possible worlds

several possible worlds

```
0.8::stress(ann).
0.6::influences(ann,bob).
0.2::influences(bob,carl).
```

atoms as random variables



Prolog / logic programming

```
stress(ann).
influences(ann,bob).
influences(bob,carl).
```

one world

```
smokes(X) :- stress(X).
smokes(X) :-
     influences(Y,X), smokes(Y).
```

parameter learning, adapted relational learning techniques

http://dtai.cs.kuleuven.be/problog/

Probabilistic Prologs: Two Views

- Distribution semantics:
  - probability distribution over interpretations
  - degree of belief
- Stochastic Logic Programs (SLPs):
  - probability distribution over query answers
  - like in probabilistic grammars

Graphs & Randomness

ProbLog, Phenetic, Prism, ICL, Probabilistic Databases, ...

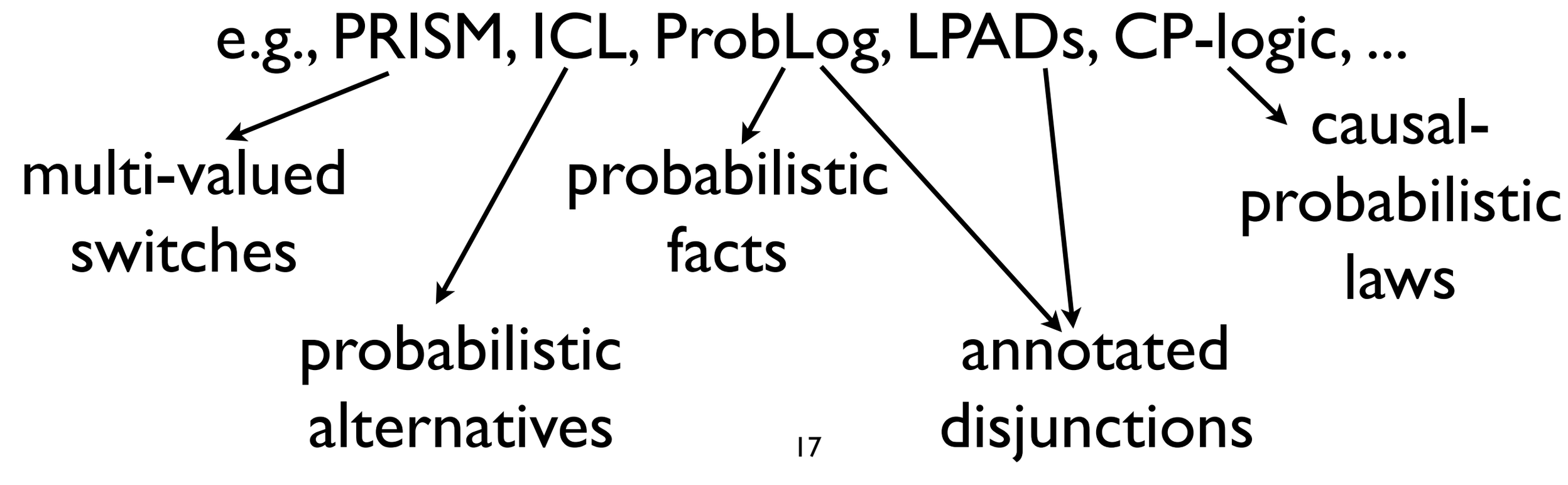
- all based on a "random graph" model

Stochastic Logic Programs, ProPPR, PCFGs, ...

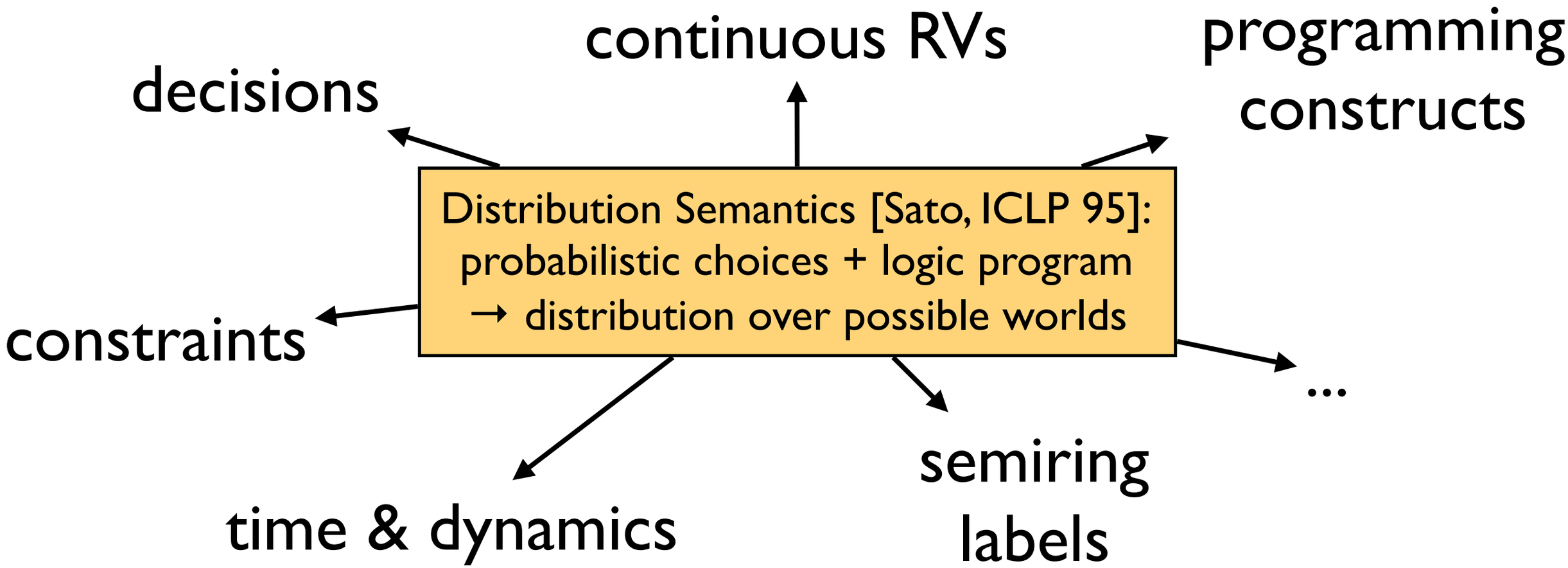
- based on a "random walk" model
- connected to PageRank

Probabilistic Logic Programming

Distribution Semantics [Sato, ICLP 95]:
probabilistic choices + logic program
→ distribution over possible worlds



Extensions of basic PLP



Roadmap

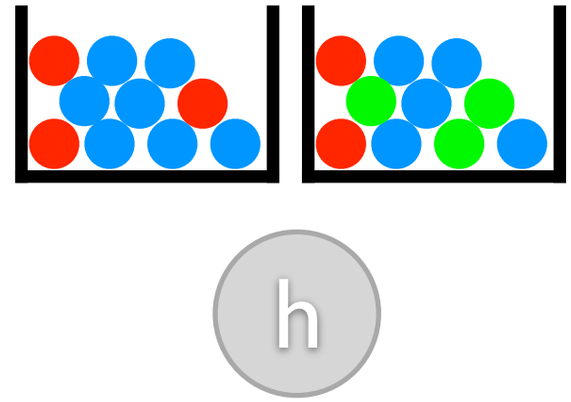
- Modeling
- Reasoning
- Language extensions
- Learning
- Advanced topics

... with some detours on the way

Part 1 : Modeling

ProbLog by example:

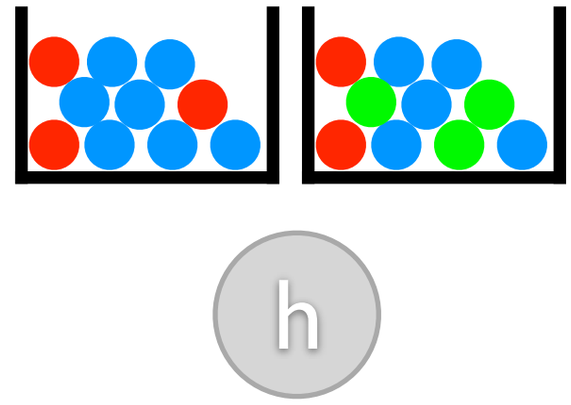
A bit of gambling



- toss (biased) coin & draw ball from each urn
- win if (heads and a red ball) or (two balls of same color)

ProbLog by example:

A bit of gambling



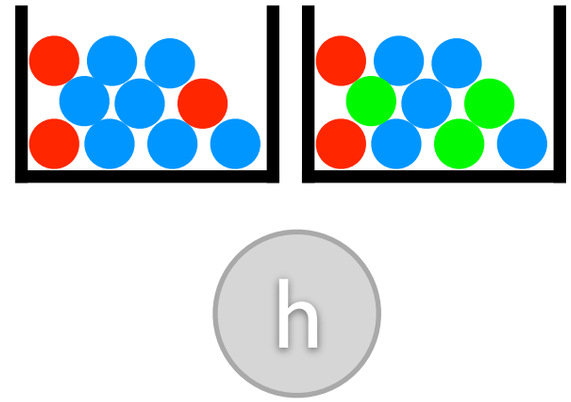
- toss (biased) coin & draw ball from each urn
- win if (heads and a red ball) or (two balls of same color)

```
0.4 :: heads.
```

probabilistic fact: heads is true with probability 0.4 (and false with 0.6)

ProbLog by example:

A bit of gambling



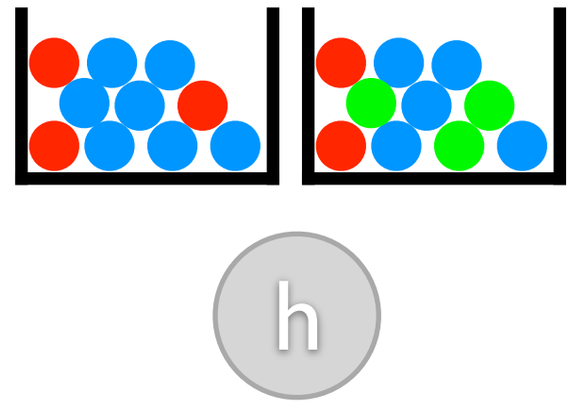
- toss (biased) coin & draw ball from each urn
- win if (heads and a red ball) or (two balls of same color)

```
0.4 :: heads.

0.3 :: col(1,red); 0.7 :: col(1,blue).
```

annotated disjunction: first ball is red with probability 0.3 and blue with 0.7

ProbLog by example:

A bit of gambling



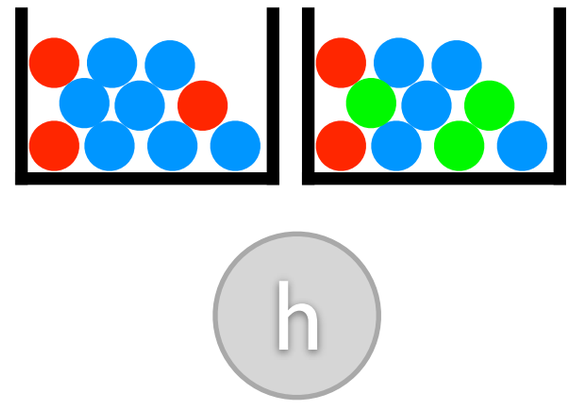
- toss (biased) coin & draw ball from each urn
- win if (heads and a red ball) or (two balls of same color)

```
0.4 :: heads.

0.3 :: col(1,red); 0.7 :: col(1,blue).
0.2 :: col(2,red); 0.3 :: col(2,green);
                   0.5 :: col(2,blue).
```

annotated disjunction: second ball is red with probability 0.2, green with 0.3, and blue with 0.5

ProbLog by example:

A bit of gambling



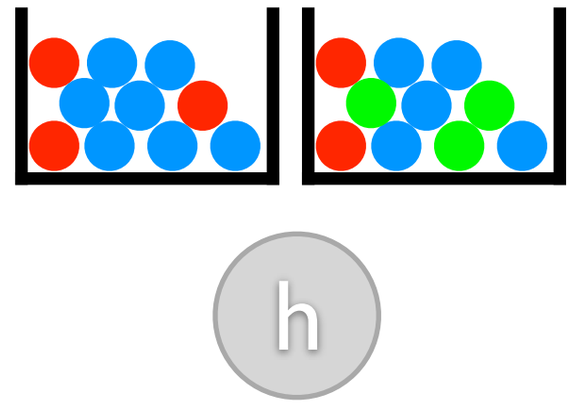
- toss (biased) coin & draw ball from each urn
- win if (heads and a red ball) or (two balls of same color)

```
0.4 :: heads.

0.3 :: col(1,red); 0.7 :: col(1,blue).
0.2 :: col(2,red); 0.3 :: col(2,green);
                   0.5 :: col(2,blue).

win :- heads, col(_,red).
```

logical rule encoding background knowledge

ProbLog by example:

A bit of gambling



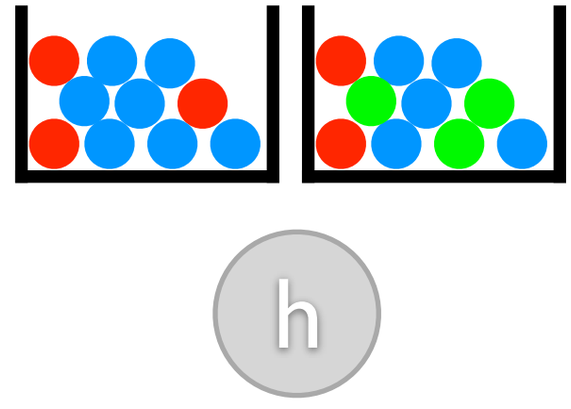
- toss (biased) coin & draw ball from each urn
- win if (heads and a red ball) or (two balls of same color)

```
0.4 :: heads.

0.3 :: col(1,red); 0.7 :: col(1,blue).
0.2 :: col(2,red); 0.3 :: col(2,green);
                   0.5 :: col(2,blue).

win :- heads, col(_,red).
win :- col(1,C), col(2,C).
```

logical rule encoding background knowledge

ProbLog by example:

A bit of gambling



- toss (biased) coin & draw ball from each urn
- win if (heads and a red ball) or (two balls of same color)

```
0.4 :: heads.

0.3 :: col(1,red); 0.7 :: col(1,blue).
0.2 :: col(2,red); 0.3 :: col(2,green);
                   0.5 :: col(2,blue).
```

probabilistic choices

```
win :- heads, col(_,red).
win :- col(1,C), col(2,C).
```

consequences

Questions

```
0.4 :: heads.

0.3 :: col(1,red); 0.7 :: col(1,blue).
0.2 :: col(2,red); 0.3 :: col(2,green); 0.5 :: col(2,blue).

win :- heads, col(_,red).
win :- col(1,C), col(2,C).
```

- Probability of **`win`**?
- Probability of **`win`** given **`col(2,green)`**?
- Most probable world where **`win`** is true?

Questions

```
0.4 :: heads.

0.3 :: col(1,red); 0.7 :: col(1,blue).
0.2 :: col(2,red); 0.3 :: col(2,green); 0.5 :: col(2,blue).

win :- heads, col(_,red).
win :- col(1,C), col(2,C).
```

marginal probability

- Probability of `win`?

query

- Probability of `win` given `col(2,green)`?
- Most probable world where `win` is true?

Questions

```
0.4 :: heads.

0.3 :: col(1,red); 0.7 :: col(1,blue).
0.2 :: col(2,red); 0.3 :: col(2,green); 0.5 :: col(2,blue).

win :- heads, col(_,red).
win :- col(1,C), col(2,C).
```

marginal probability

- Probability of **`win`**?

conditional probability

- Probability of **`win`** given **`col(2,green)`**?

evidence

- Most probable world where **`win`** is true?

Questions

```
0.4 :: heads.

0.3 :: col(1,red); 0.7 :: col(1,blue).
0.2 :: col(2,red); 0.3 :: col(2,green); 0.5 :: col(2,blue).

win :- heads, col(_,red).
win :- col(1,C), col(2,C).
```

marginal probability

- Probability of **`win`**?

conditional probability

- Probability of **`win`** given **`col(2,green)`**?

- Most probable world where **`win`** is true?

MPE inference

Possible Worlds

```
0.4 :: heads.

0.3 :: col(1,red); 0.7 :: col(1,blue).
0.2 :: col(2,red); 0.3 :: col(2,green); 0.5 :: col(2,blue).

win :- heads, col( ,red).
win :- col(1,C), col(2,C).
```

Possible Worlds

```
0.4 :: heads.

0.3 :: col(1,red); 0.7 :: col(1,blue).
0.2 :: col(2,red); 0.3 :: col(2,green); 0.5 :: col(2,blue).

win :- heads, col( ,red).
win :- col(1,C), col(2,C).
```

Possible Worlds

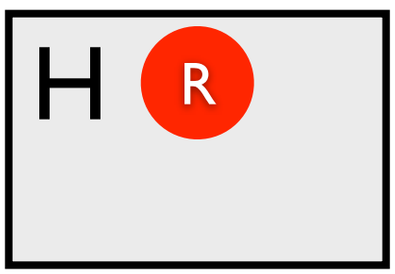
```
0.4 :: heads.

0.3 :: col(1,red); 0.7 :: col(1,blue).
0.2 :: col(2,red); 0.3 :: col(2,green); 0.5 :: col(2,blue).

win :- heads, col(_,red).
win :- col(1,C), col(2,C).
```

0.4

Possible Worlds

```
0.4 :: heads.

0.3 :: col(1,red); 0.7 :: col(1,blue).
0.2 :: col(2,red); 0.3 :: col(2,green); 0.5 :: col(2,blue).

win :- heads, col( ,red).
win :- col(1,C), col(2,C).
```

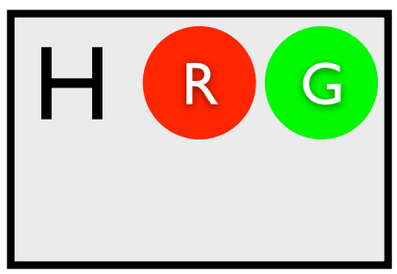
0.4 ×0.3



Possible Worlds

```
0.4 :: heads.

0.3 :: col(1,red); 0.7 :: col(1,blue).
0.2 :: col(2,red); 0.3 :: col(2,green); 0.5 :: col(2,blue).

win :- heads, col(_,red).
win :- col(1,C), col(2,C).
```

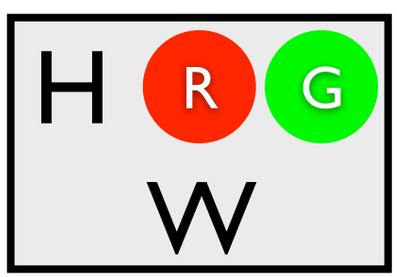
0.4 ×0.3 ×0.3



Possible Worlds

```
0.4 :: heads.

0.3 :: col(1,red); 0.7 :: col(1,blue).
0.2 :: col(2,red); 0.3 :: col(2,green); 0.5 :: col(2,blue).

win :- heads, col(_,red).
win :- col(1,C), col(2,C).
```

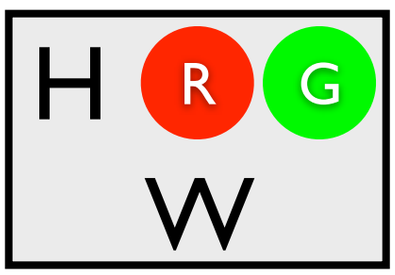
0.4 ×0.3 ×0.3



Possible Worlds

```
0.4 :: heads.

0.3 :: col(1,red); 0.7 :: col(1,blue).
0.2 :: col(2,red); 0.3 :: col(2,green); 0.5 :: col(2,blue).

win :- heads, col(_,red).
win :- col(1,C), col(2,C).
```

$0.4 \times 0.3 \times 0.3$

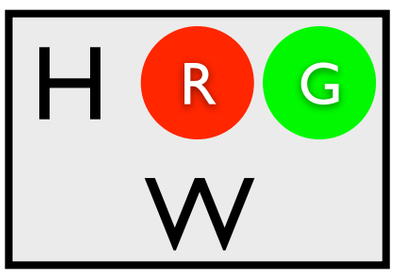


$(1-0.4)$

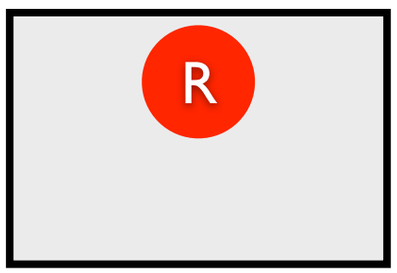
Possible Worlds

```
0.4 :: heads.

0.3 :: col(1,red); 0.7 :: col(1,blue).
0.2 :: col(2,red); 0.3 :: col(2,green); 0.5 :: col(2,blue).

win :- heads, col(_,red).
win :- col(1,C), col(2,C).
```

0.4 ×0.3 ×0.3



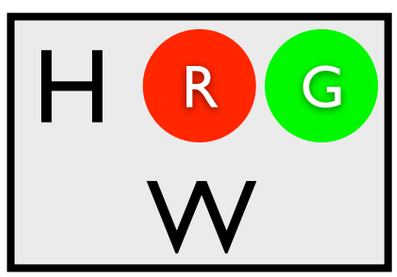
(1−0.4)×0.3



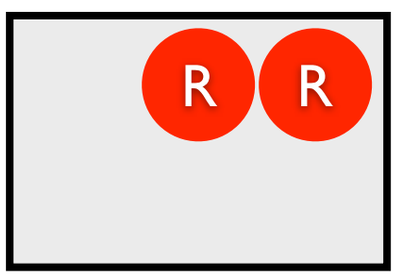
Possible Worlds

```
0.4 :: heads.

0.3 :: col(1,red); 0.7 :: col(1,blue).
0.2 :: col(2,red); 0.3 :: col(2,green); 0.5 :: col(2,blue).

win :- heads, col(_,red).
win :- col(1,C), col(2,C).
```

0.4 ×0.3 ×0.3



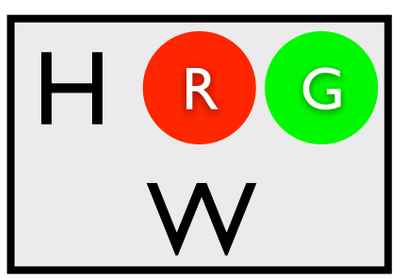
(1−0.4)×0.3 ×0.2



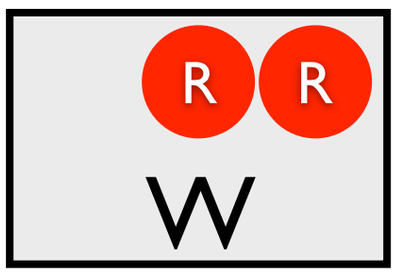
Possible Worlds

```
0.4 :: heads.

0.3 :: col(1,red); 0.7 :: col(1,blue).
0.2 :: col(2,red); 0.3 :: col(2,green); 0.5 :: col(2,blue).

win :- heads, col(_,red).
win :- col(1,C), col(2,C).
```

$0.4 \times 0.3 \times 0.3$



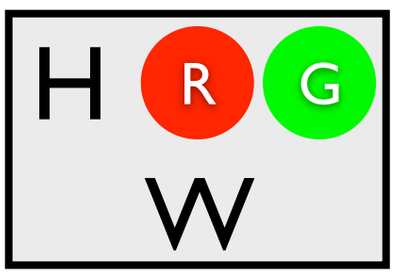
$(1-0.4) \times 0.3 \times 0.2$



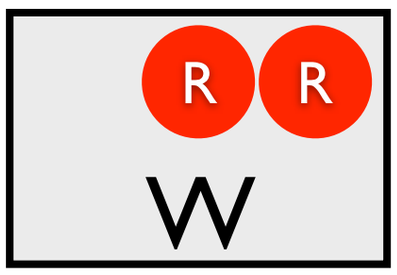
Possible Worlds

```
0.4 :: heads.

0.3 :: col(1,red); 0.7 :: col(1,blue).
0.2 :: col(2,red); 0.3 :: col(2,green); 0.5 :: col(2,blue).

win :- heads, col(_,red).
win :- col(1,C), col(2,C).
```

0.4 ×0.3 ×0.3



(1−0.4)×0.3 ×0.2

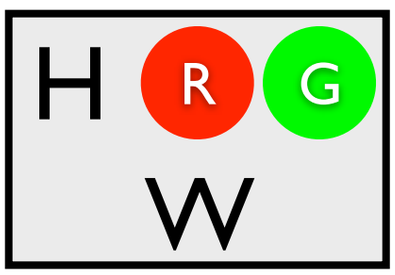


(1−0.4)

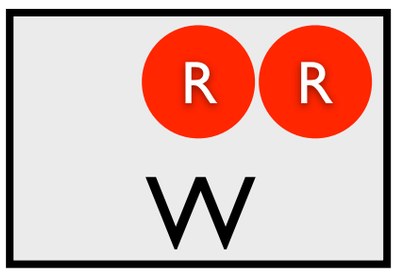
Possible Worlds

```
0.4 :: heads.

0.3 :: col(1,red); 0.7 :: col(1,blue).
0.2 :: col(2,red); 0.3 :: col(2,green); 0.5 :: col(2,blue).

win :- heads, col(_,red).
win :- col(1,C), col(2,C).
```

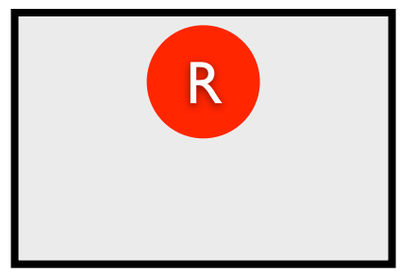
$0.4 \times 0.3 \times 0.3$



$(1-0.4) \times 0.3 \times 0.2$



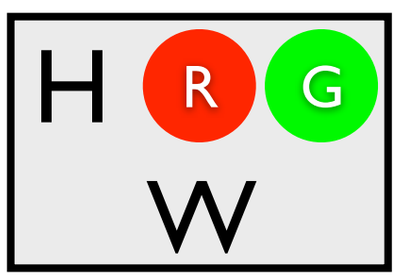
$(1-0.4) \times 0.3$



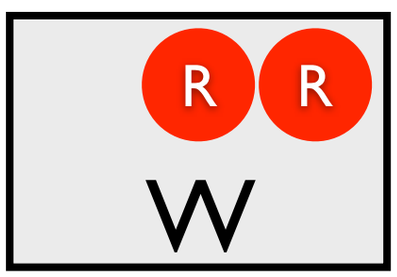
Possible Worlds

```
0.4 :: heads.

0.3 :: col(1,red); 0.7 :: col(1,blue).
0.2 :: col(2,red); 0.3 :: col(2,green); 0.5 :: col(2,blue).

win :- heads, col(_,red).
win :- col(1,C), col(2,C).
```

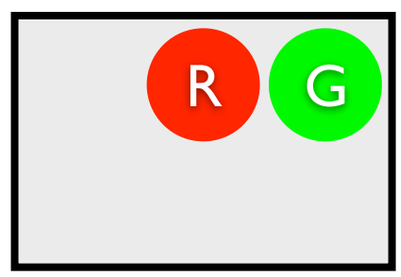
$0.4 \times 0.3 \times 0.3$



$(1-0.4) \times 0.3 \times 0.2$



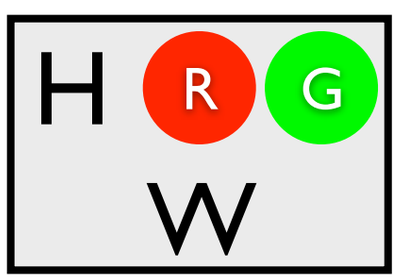
$(1-0.4) \times 0.3 \times 0.3$



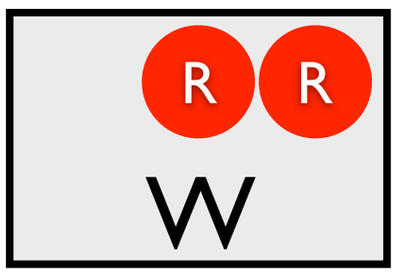
Possible Worlds

```
0.4 :: heads.

0.3 :: col(1,red); 0.7 :: col(1,blue).
0.2 :: col(2,red); 0.3 :: col(2,green); 0.5 :: col(2,blue).

win :- heads, col(_,red).
win :- col(1,C), col(2,C).
```

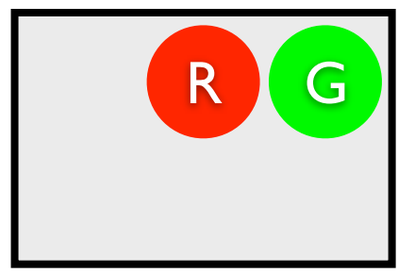
$0.4 \times 0.3 \times 0.3$



$(1-0.4) \times 0.3 \times 0.2$



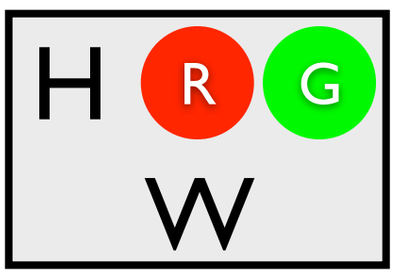
$(1-0.4) \times 0.3 \times 0.3$



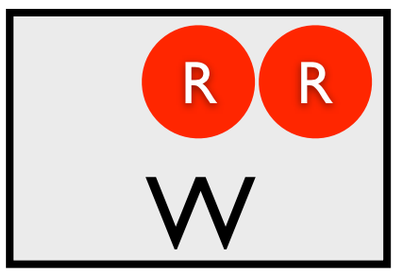
Possible Worlds

```
0.4 :: heads.

0.3 :: col(1,red); 0.7 :: col(1,blue).
0.2 :: col(2,red); 0.3 :: col(2,green); 0.5 :: col(2,blue).

win :- heads, col( ,red).
win :- col(1,C), col(2,C).
```

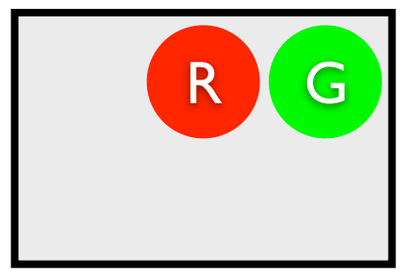
$0.4 \times 0.3 \times 0.3$



$(1-0.4) \times 0.3 \times 0.2$

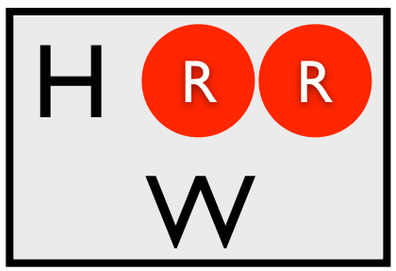


$(1-0.4) \times 0.3 \times 0.3$

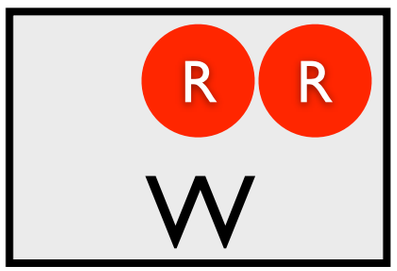


All Possible Worlds

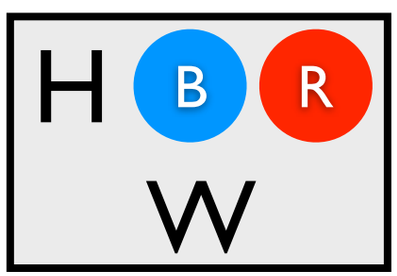
0.024



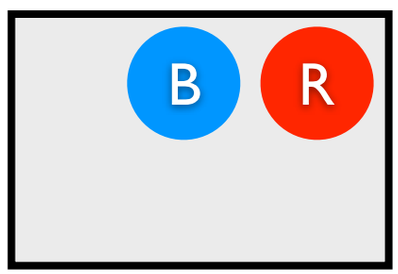
0.036



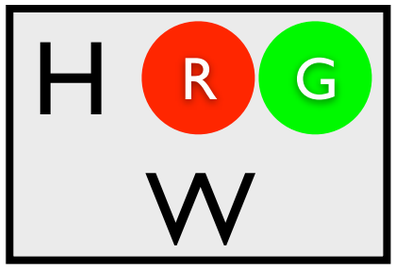
0.056



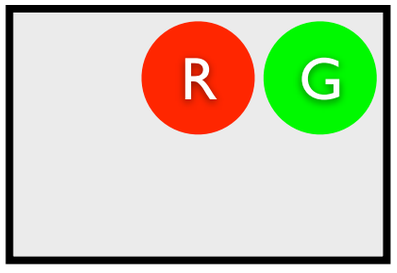
0.084



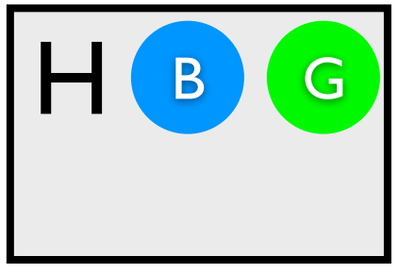
0.036



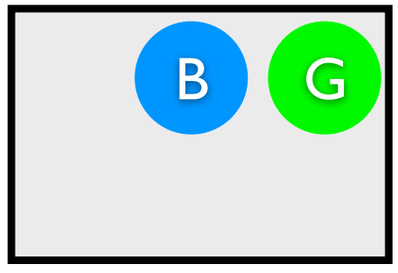
0.054



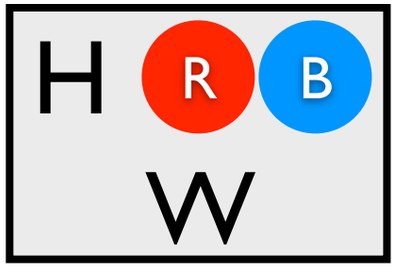
0.084



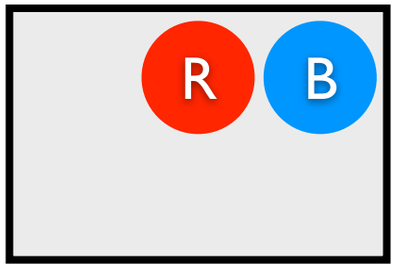
0.126



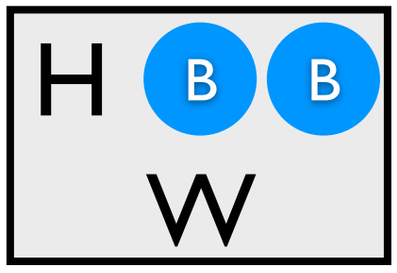
0.060



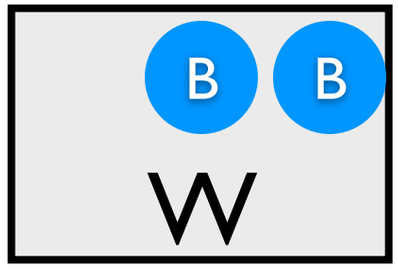
0.090



0.140

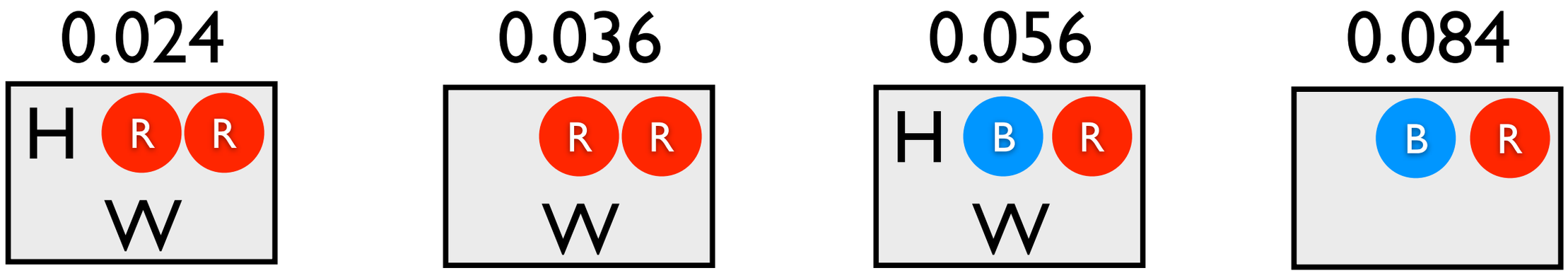


0.210



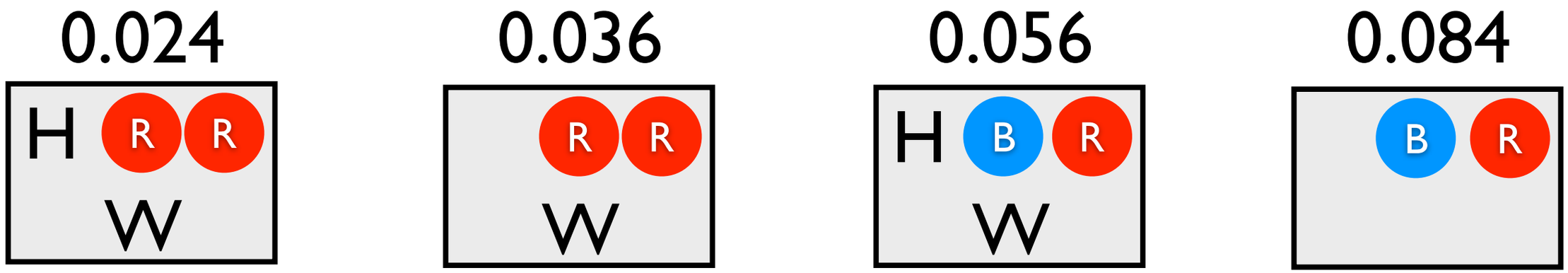
Most likely world where **win** is true?

MPE Inference



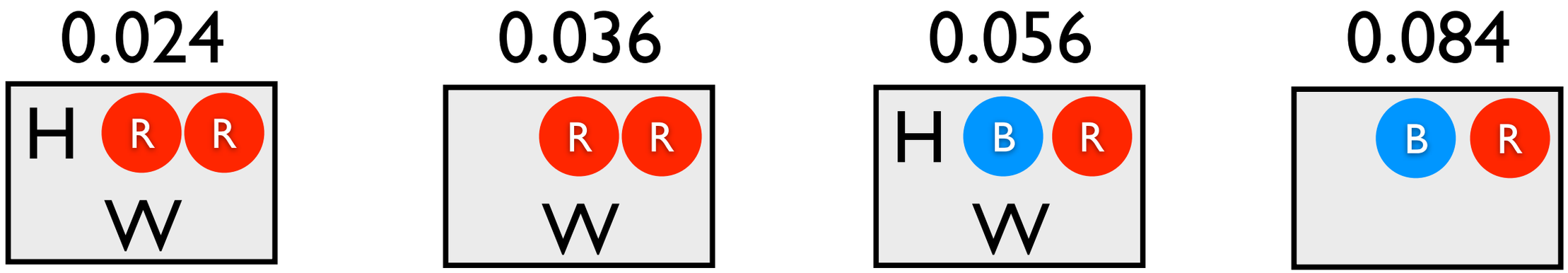
Most likely world where **win** is true?

MPE Inference



Most likely world where `col(2,blue)` is false?

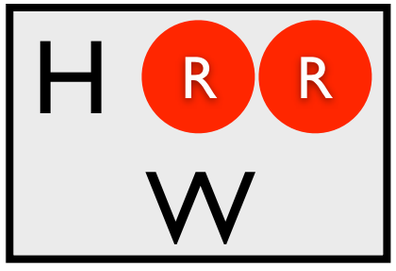
MPE Inference



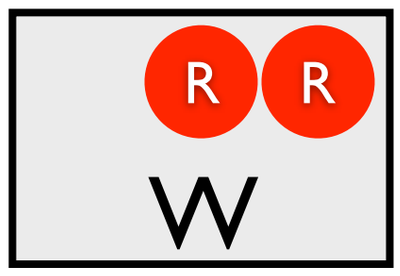
Most likely world where `col(2,blue)` is false?

MPE Inference

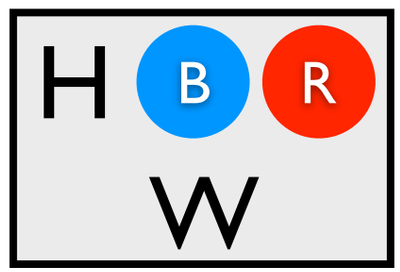
0.024



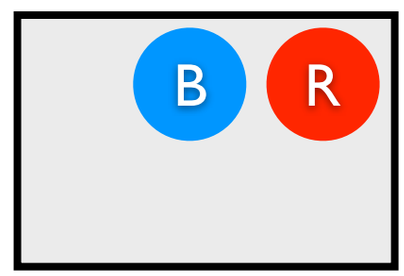
0.036



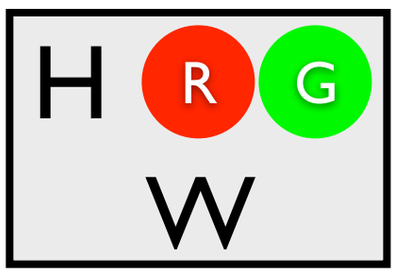
0.056



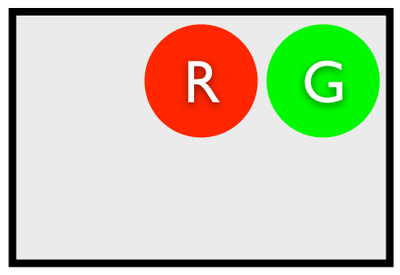
0.084



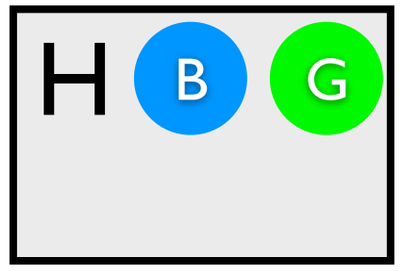
0.036



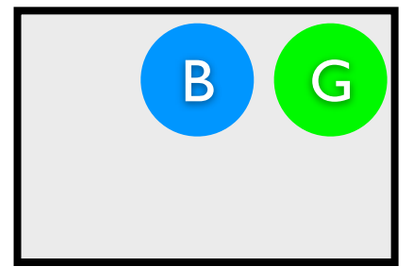
0.054



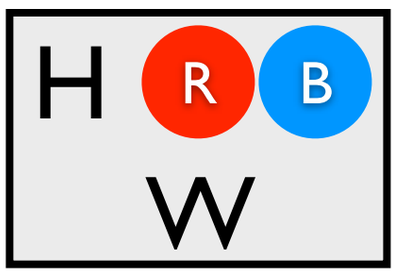
0.084



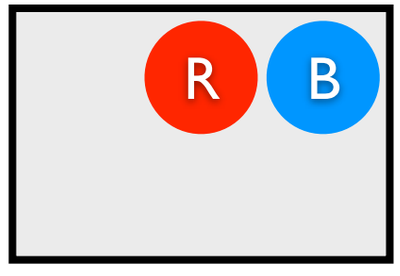
0.126



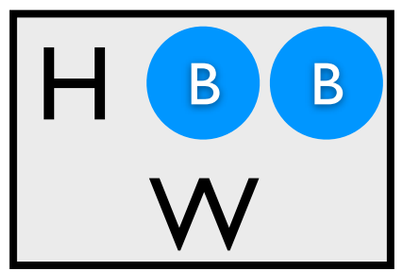
0.060



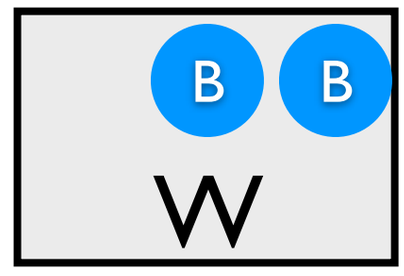
0.090



0.140



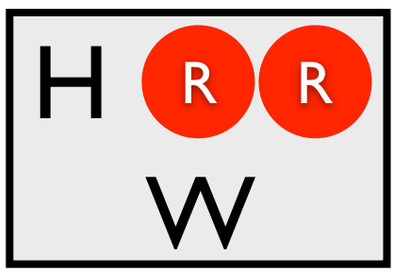
0.210



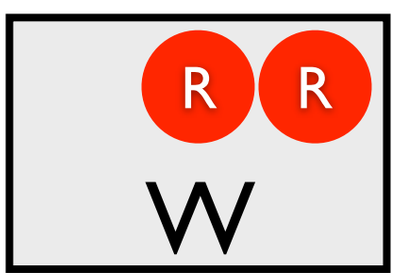
P(win)= ?

Marginal Probability

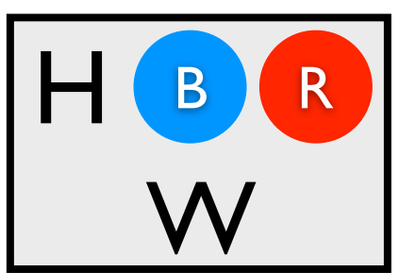
0.024



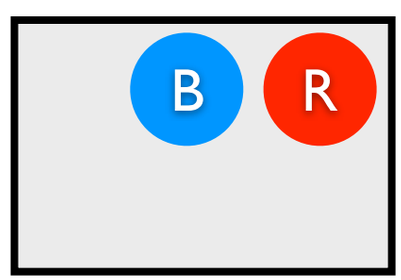
0.036



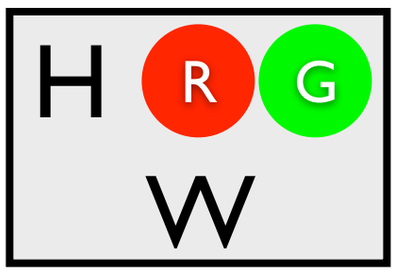
0.056



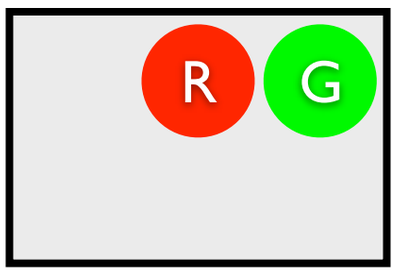
0.084



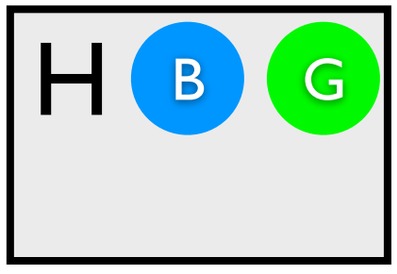
0.036



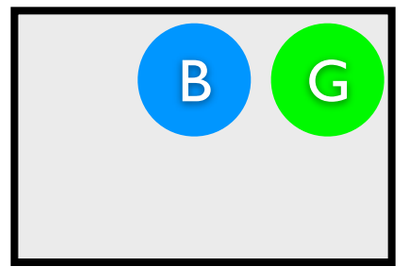
0.054



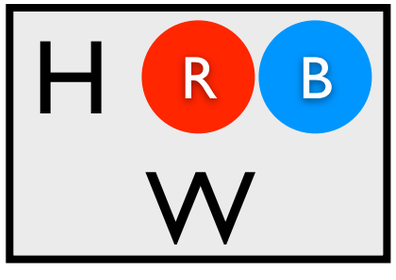
0.084



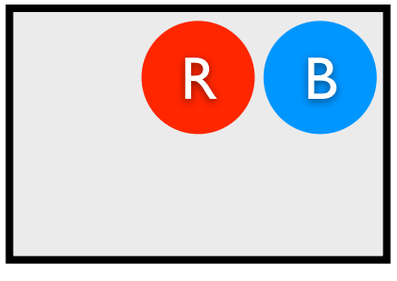
0.126



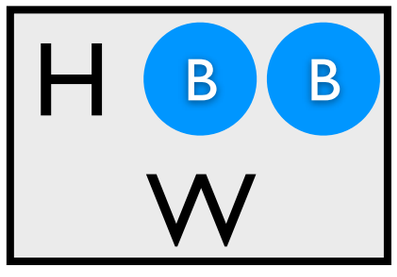
0.060



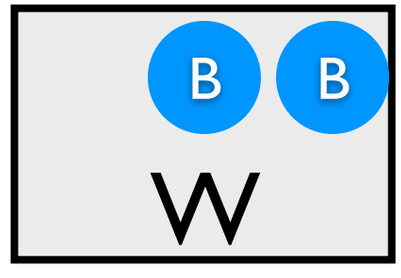
0.090



0.140



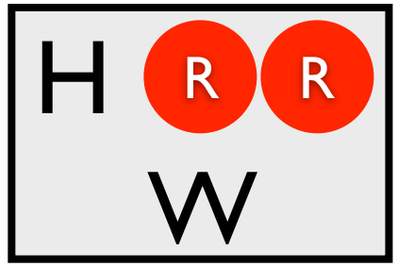
0.210



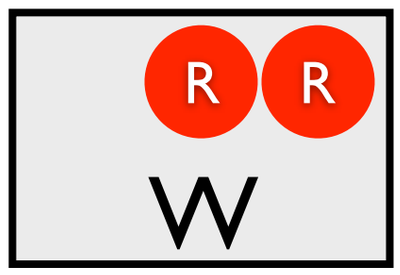
P(win)= Σ

Marginal Probability

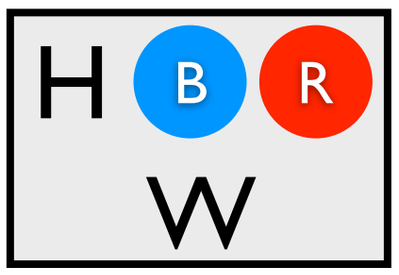
0.024



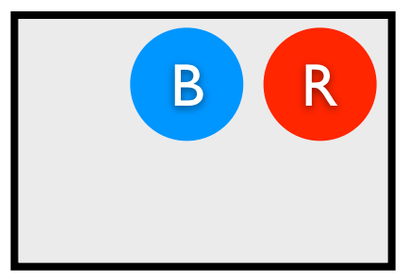
0.036



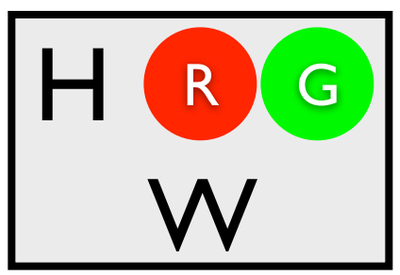
0.056



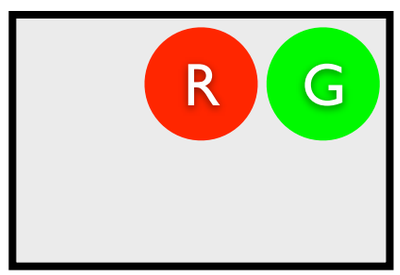
0.084



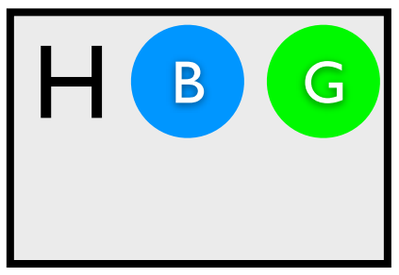
0.036



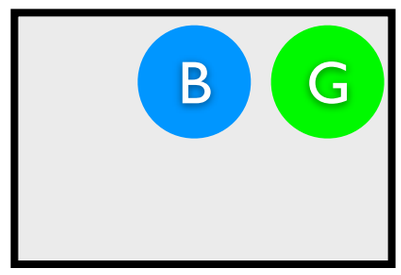
0.054



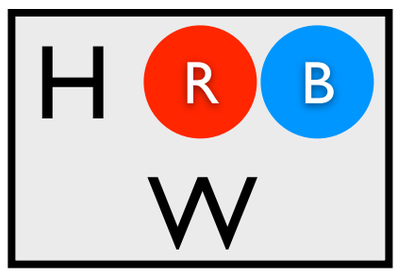
0.084



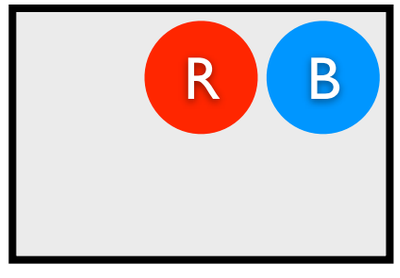
0.126



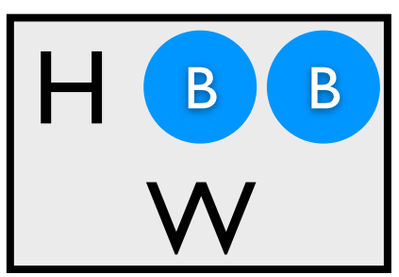
0.060



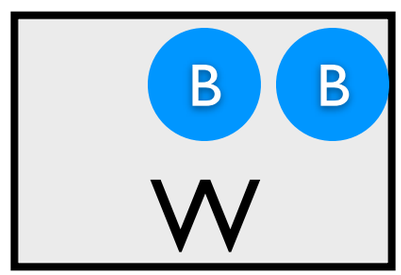
0.090



0.140



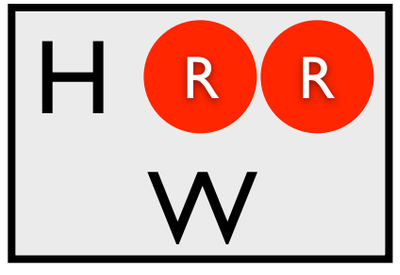
0.210



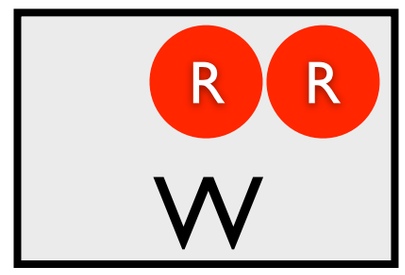
P(win)= $\sum$ =0.562

Marginal Probability

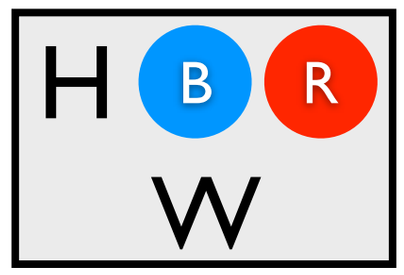
0.024



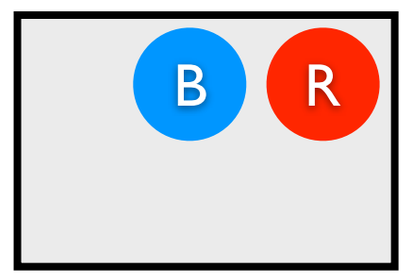
0.036



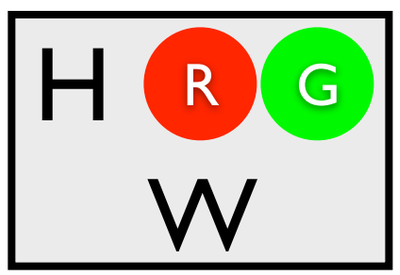
0.056



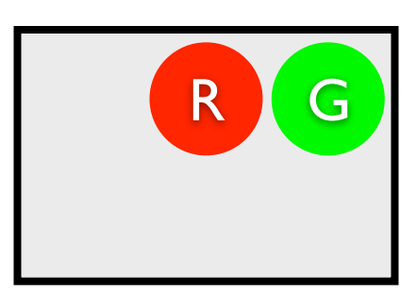
0.084



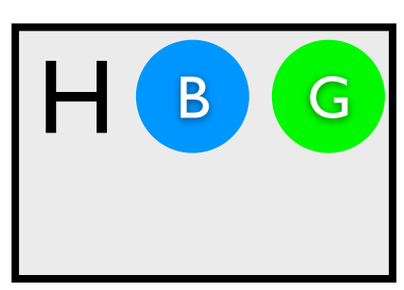
0.036



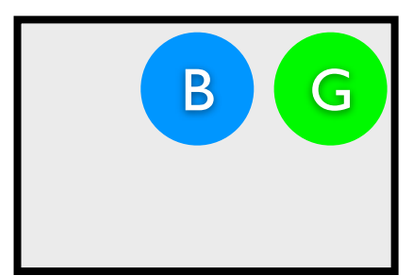
0.054



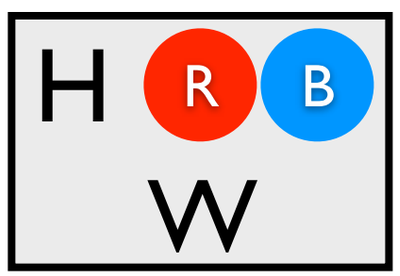
0.084



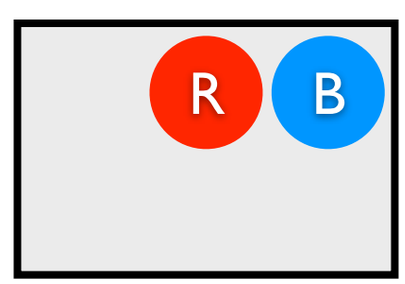
0.126



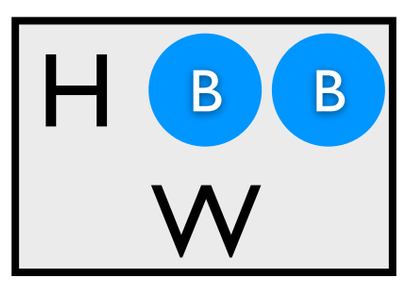
0.060



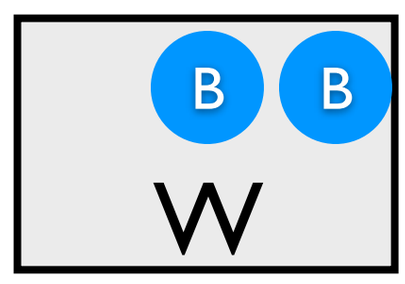
0.090



0.140



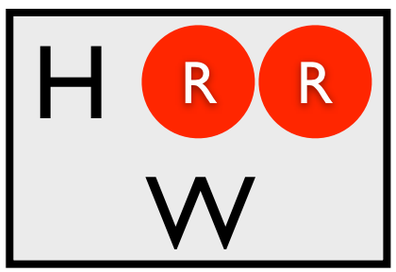
0.210



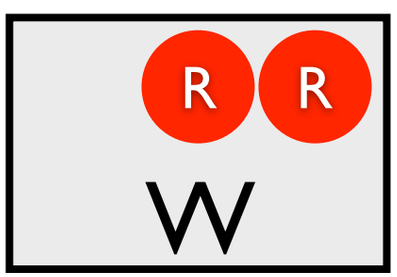
P(win|col(2,green))= ?

Conditional Probability

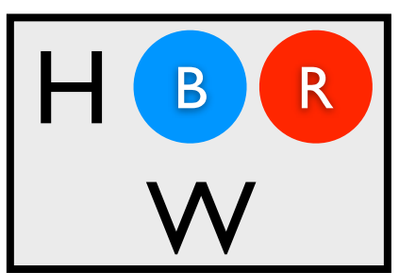
0.024



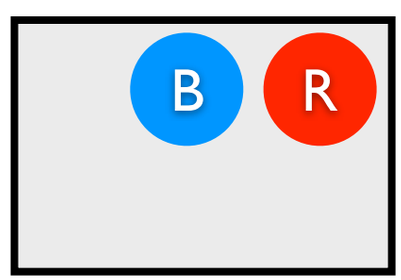
0.036



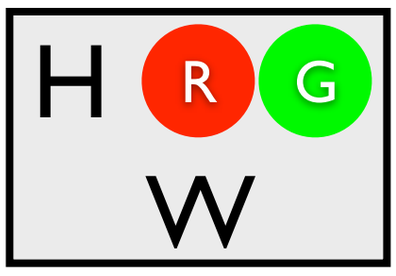
0.056



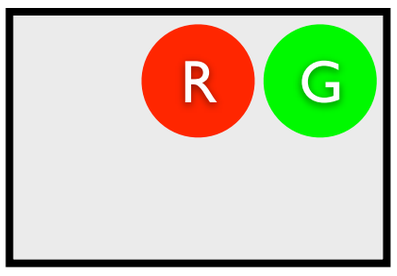
0.084



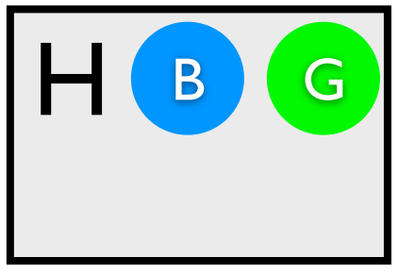
0.036



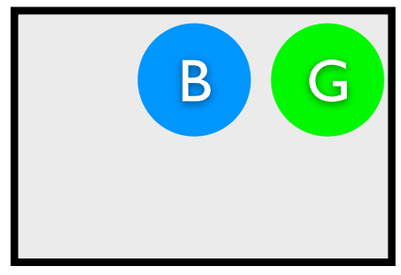
0.054



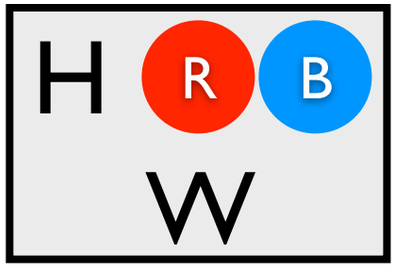
0.084



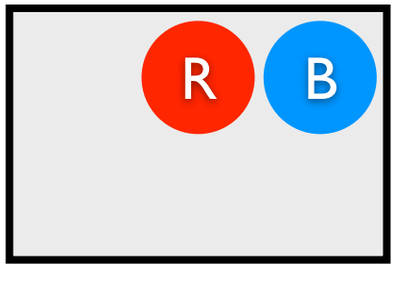
0.126



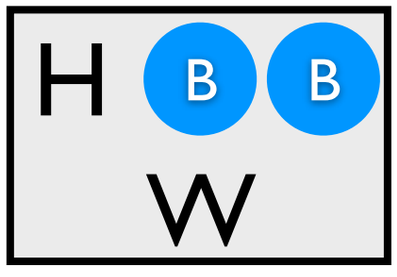
0.060



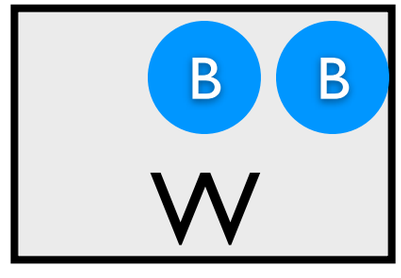
0.090



0.140



0.210

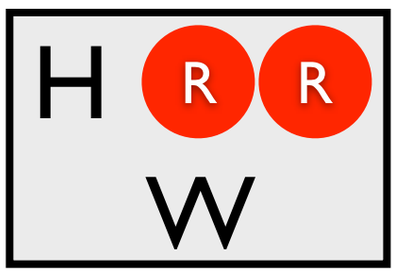


$$P(win|col(2,green)) = \Sigma/\Sigma$$

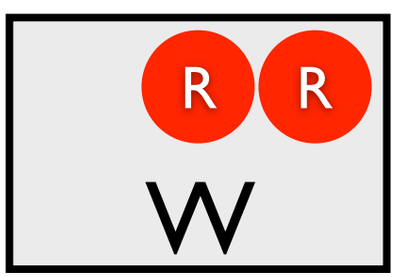
$$= P(win \wedge col(2,green))/P(col(2,green))$$

Conditional Probability

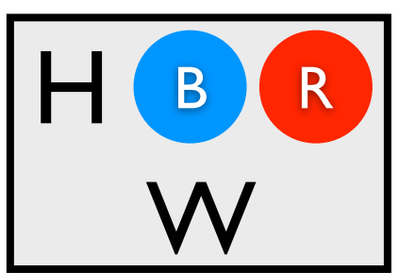
0.024



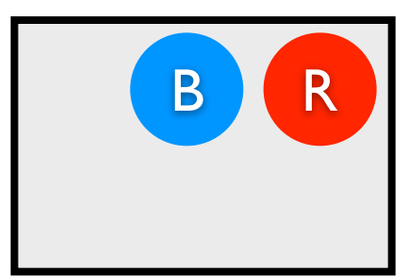
0.036



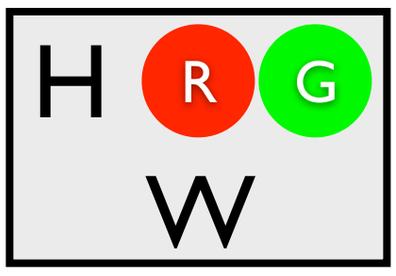
0.056



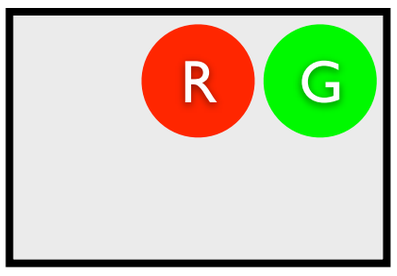
0.084



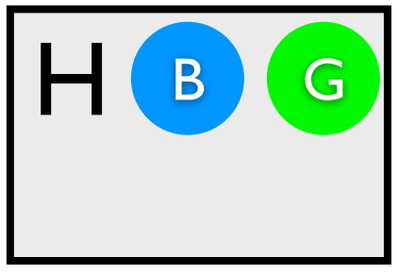
0.036



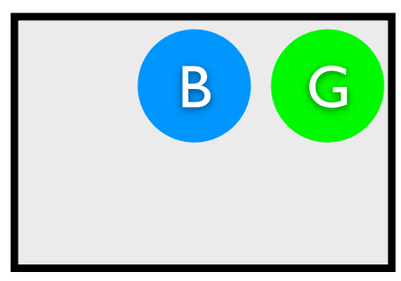
0.054



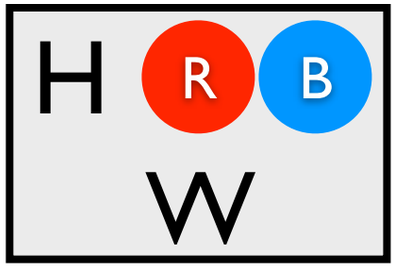
0.084



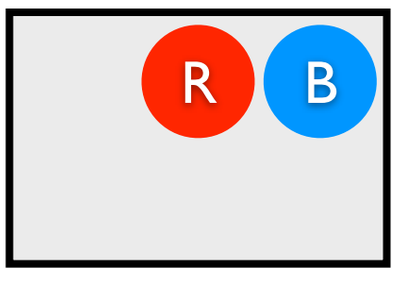
0.126



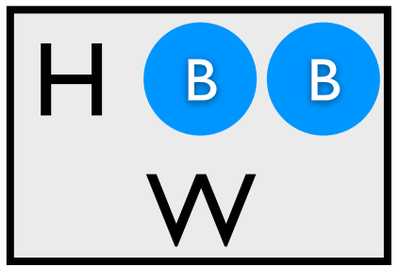
0.060



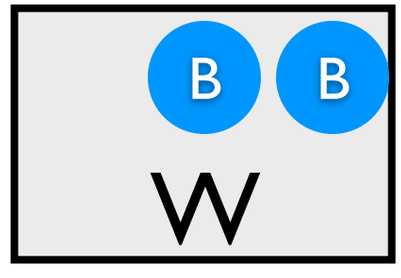
0.090



0.140



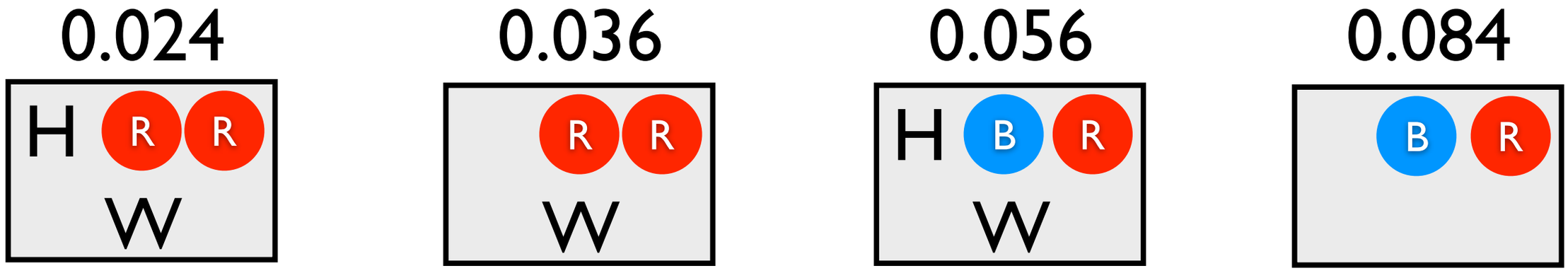
0.210



$$P(win|col(2,green)) = \Sigma/\Sigma$$

$$= P(win \wedge col(2,green))/P(col(2,green))$$

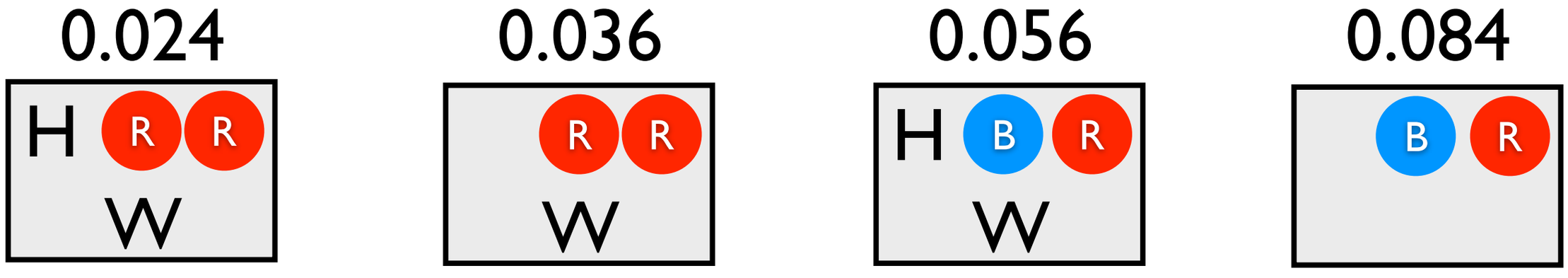
Conditional Probability



$$P(win|col(2,green)) = \Sigma/\Sigma$$

$$= 0.036/0.3 = 0.12$$

Conditional Probability



Distribution Semantics

(with probabilistic facts)

[Sato, ICLP 95]

$$P(Q) = \sum_{F \cup R \models Q} \prod_{f \in F} p(f) \prod_{f \notin F} 1 - p(f)$$

Distribution Semantics

(with probabilistic facts)

[Sato, ICLP 95]

query

$$P(Q) = \sum_{F \cup R \models Q} \prod_{f \in F} p(f) \prod_{f \notin F} 1 - p(f)$$

Distribution Semantics

(with probabilistic facts)

[Sato, ICLP 95]

query

$$P(Q) = \sum_{F \cup R \models Q} \prod_{f \in F} p(f) \prod_{f \notin F} 1 - p(f)$$

subset of probabilistic facts

Distribution Semantics

(with probabilistic facts)

[Sato, ICLP 95]

query

$$P(Q) = \sum_{F \cup R \models Q} \prod_{f \in F} p(f) \prod_{f \notin F} 1 - p(f)$$

subset of probabilistic facts

Prolog rules

Distribution Semantics

(with probabilistic facts)

[Sato, ICLP 95]

query

sum over possible worlds where Q is true

$$P(Q) = \sum_{F \cup R \models Q} \prod_{f \in F} p(f) \prod_{f \notin F} 1 - p(f)$$

subset of probabilistic facts

Prolog rules

Distribution Semantics

(with probabilistic facts)

[Sato, ICLP 95]

query

sum over possible worlds where Q is true

$$P(Q) = \sum_{F \cup R \models Q} \prod_{f \in F} p(f) \prod_{f \notin F} 1 - p(f)$$

subset of probabilistic facts

Prolog rules

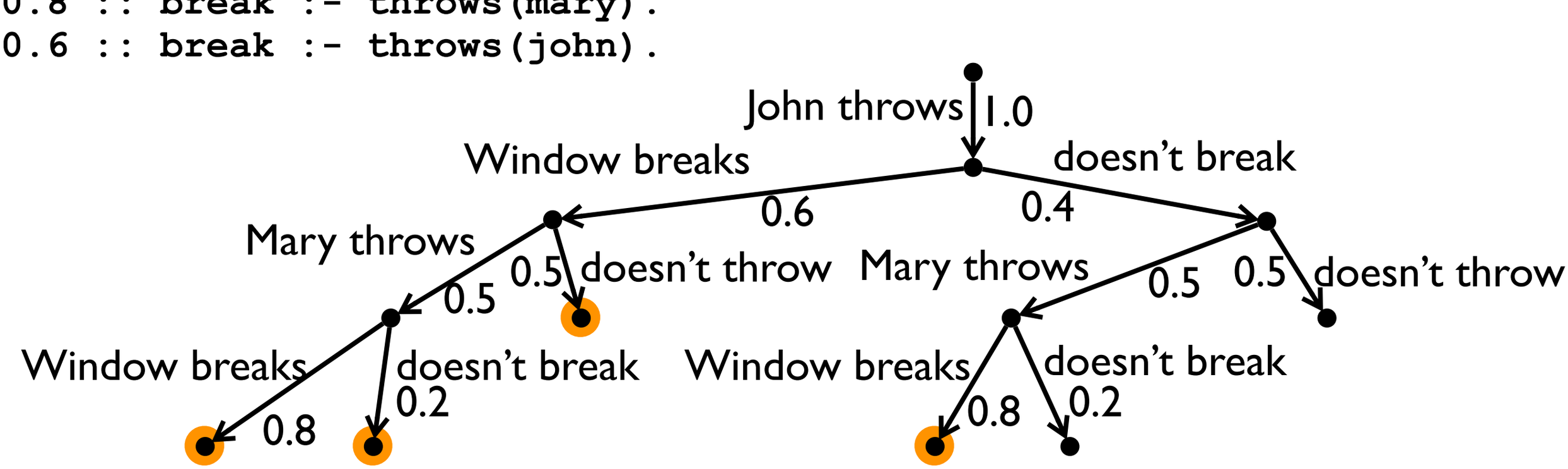
probability of possible world

[Vennekens et al, ICLP 04]

Alternative view: CP-Logic

```
throws(john).
0.5::throws(mary).

0.8 :: break :- throws(mary).
0.6 :: break :- throws(john).
```

probabilistic causal laws



P(break)=0.6×0.5×0.8+0.6×0.5×0.2+0.6×0.5+0.4×0.5×0.8

CP-logic [Vennekens et al.]

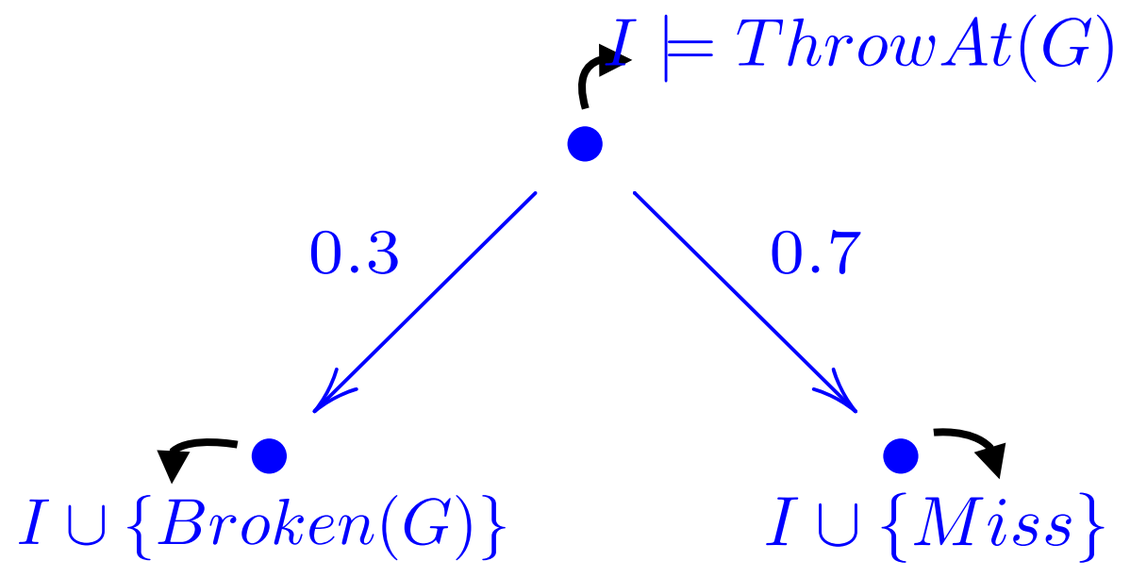
E.g., "throwing a rock at a glass breaks it with probability 0.3 and misses it with probability 0.7"

(Broken(G):0.3) ∨ (Miss 0.7) ← ThrowAt(G).

Note that the actual non-deterministic event ("rock flying at glass") is implicit

Slides CP-logic courtesy Joost Vennekens

Semantics



(Broken(G) 0.3) ∨ (Miss 0.7)
← ThrowAt(G)

Probability tree is an execution model of theory iff:

- Each tree-transition matches causal law
- The tree cannot be extended

Each execution model defines the same probability distribution over final states

Slides CP-logic courtesy Joost Vennekens

Distributional Clauses (DC)

- Discrete- and continuous-valued random variables

random variable with Gaussian distribution

```
length(Obj) ~ gaussian(6.0,0.45) :- type(Obj,glass).
stackable(OBot,OTop) :-
      ≃length(OBot) ≥ ≃length(OTop),
      ≃width(OBot) ≥ ≃width(OTop).
```

comparing values of random variables

[Gutmann et al, TPLP 11; Nitti et al, IROS 13]

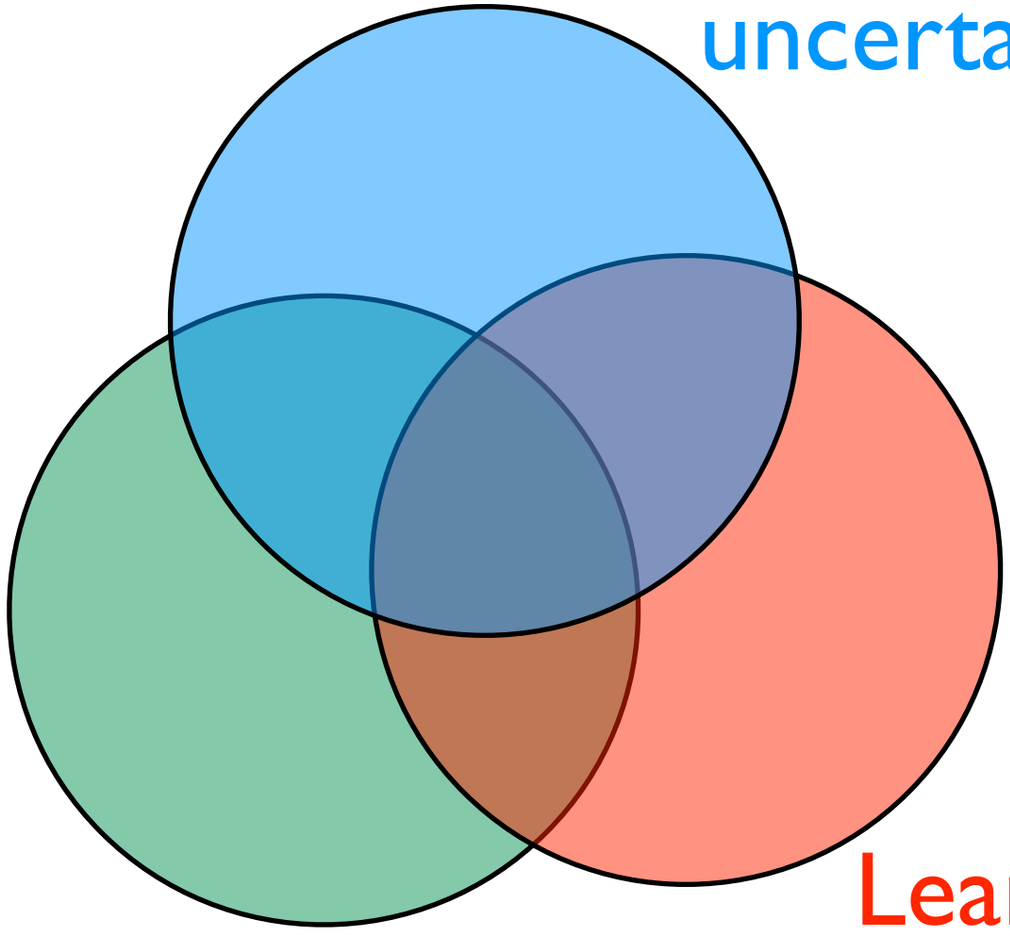
[Suciu et al 2011]

Probabilistic Databases

Dealing with uncertainty

Reasoning with relational data

Learning



[Suciu et al 2011]

Probabilistic Databases

```
select x.person, y.country
from bornIn x, cityIn y
where x.city=y.city
```

bornIn

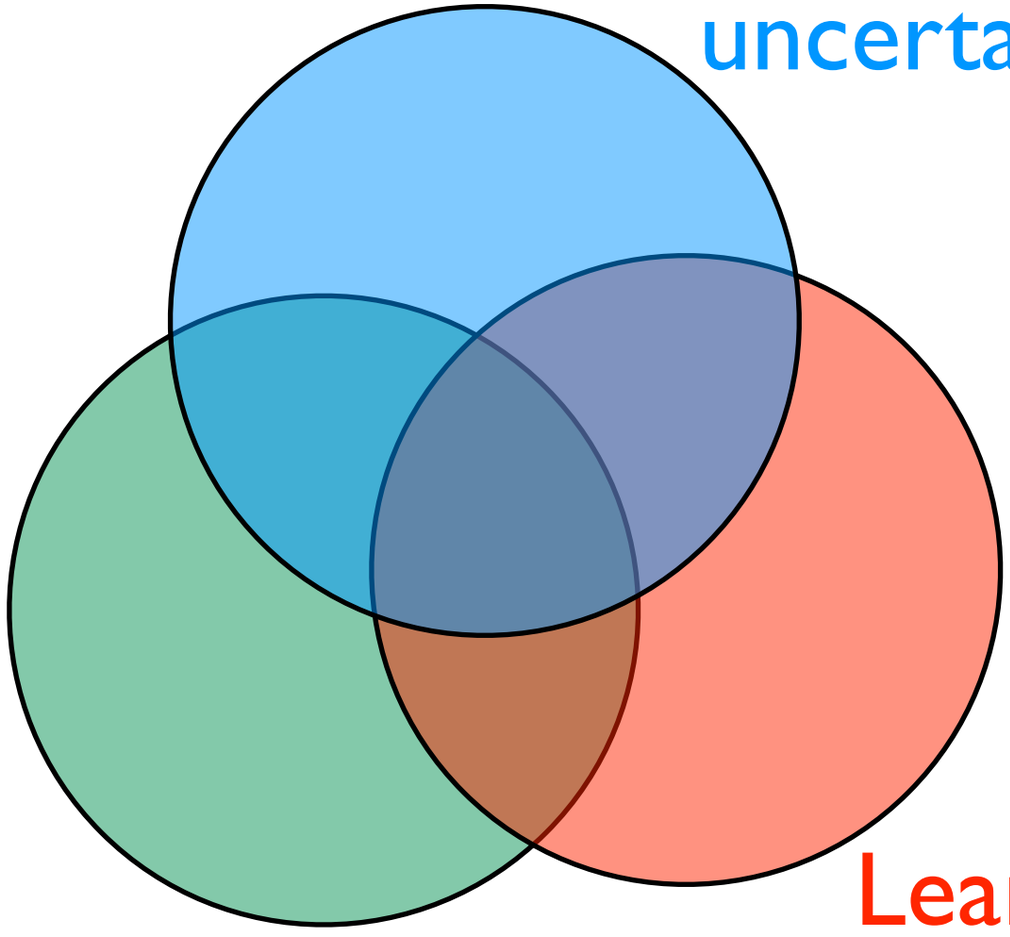
person	city
ann	london
bob	york
eve	new york
tom	paris

cityIn

city	country
london	uk
york	uk
paris	usa

relational database

Dealing with uncertainty



Learning

[Suciu et al 2011]

Probabilistic Databases

```
select x.person, y.country
from bornIn x, cityIn y
where x.city=y.city
```

one world

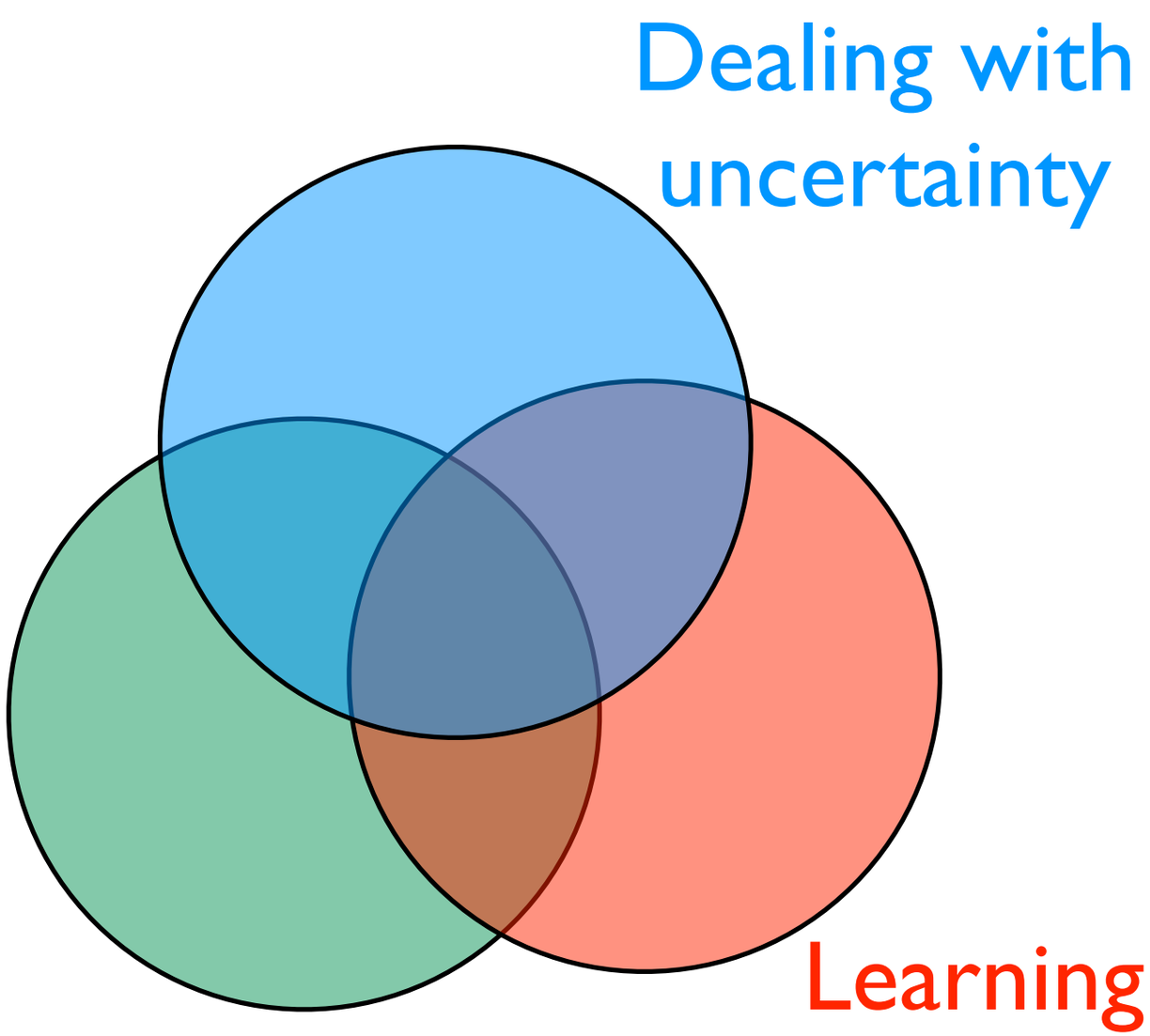
bornIn

person	city
ann	london
bob	york
eve	new york
tom	paris

cityIn

city	country
london	uk
york	uk
paris	usa

relational database



[Suciu et al 2011]

Probabilistic Databases

bornIn

person	city	P
ann	london	0,87
bob	york	0,95
eve	new york	0,9
tom	paris	0,56

cityIn

city	country	P
london	uk	0,99
york	uk	0,75
paris	usa	0,4

tuples as random variables

```
select x.person, y.country
from bornIn x, cityIn y
where x.city=y.city
```

one world

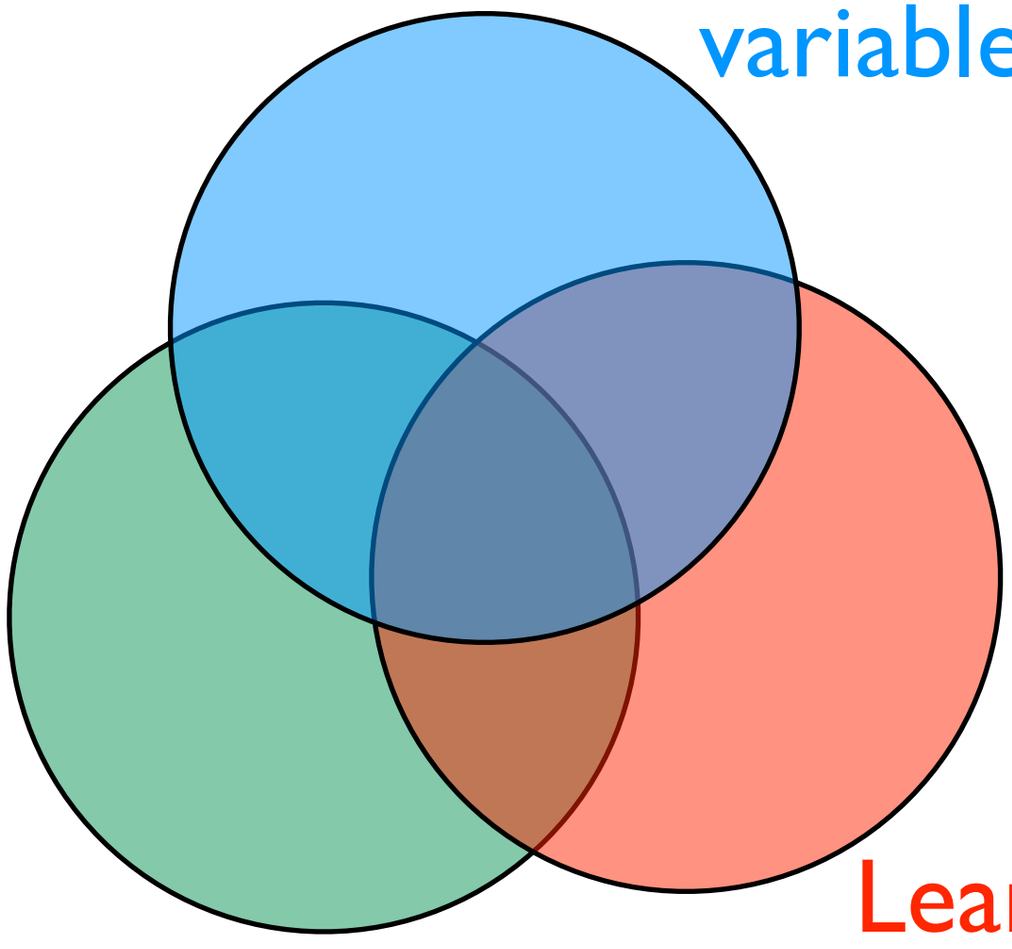
bornIn

person	city
ann	london
bob	york
eve	new york
tom	paris

cityIn

city	country
london	uk
york	uk
paris	usa

relational database



Learning

[Suciu et al 2011]

Probabilistic Databases

several possible worlds

bornIn

person	city	P
ann	london	0,87
bob	york	0,95
eve	new york	0,9
tom	paris	0,56

cityIn

city	country	P
london	uk	0,99
york	uk	0,75
paris	usa	0,4

tuples as random

```
select x.person, y.country
from bornIn x, cityIn y
where x.city=y.city
```

one world

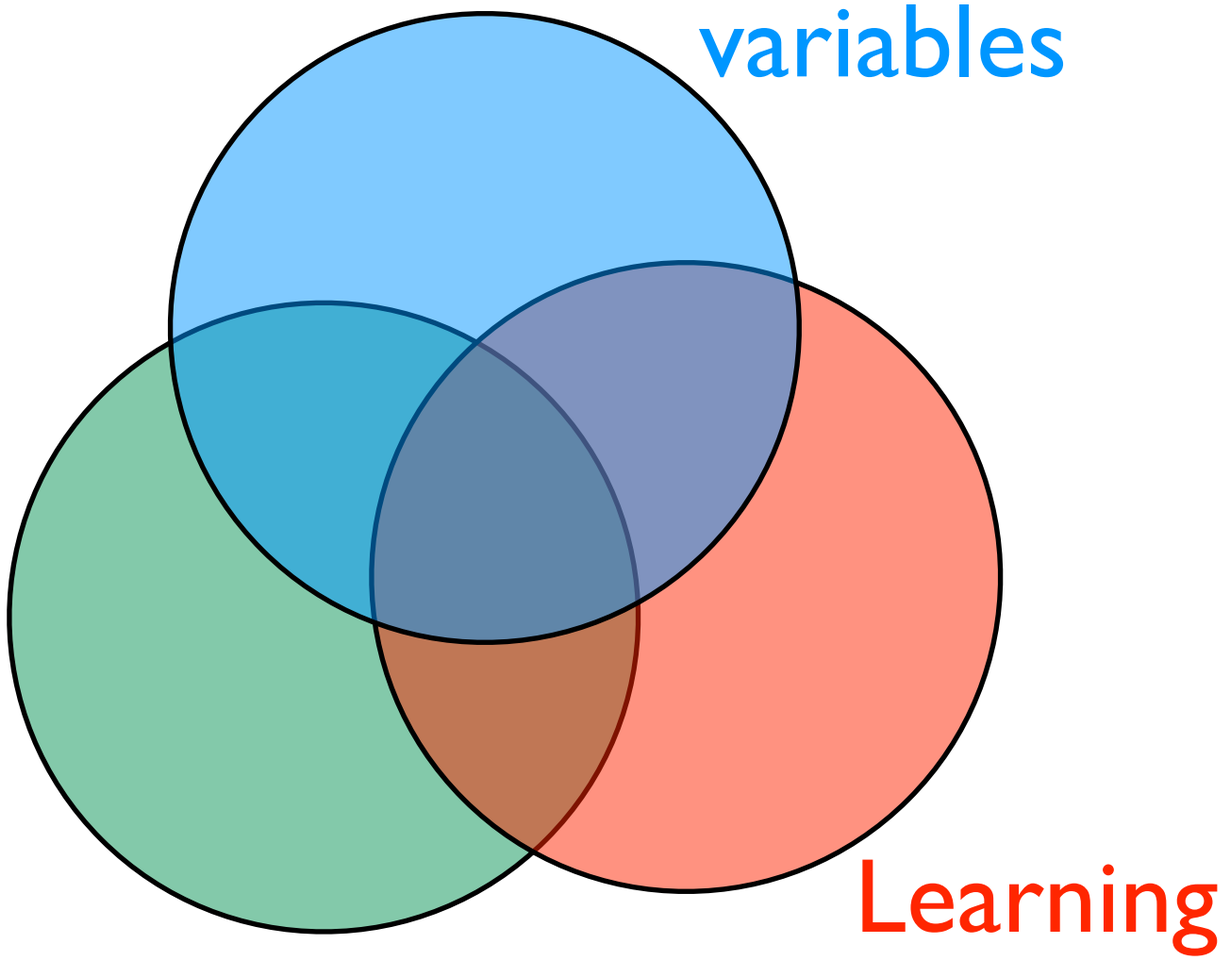
bornIn

person	city
ann	london
bob	york
eve	new york
tom	paris

cityIn

city	country
london	uk
york	uk
paris	usa

relational database



[Suciu et al 2011]

Probabilistic Databases

several possible worlds

bornIn

person	city	P
ann	london	0,87
bob	york	0,95
		0,9
		0,56

cityIn

city	country	P
london	uk	0,99
york	uk	0,75
paris	usa	0,4

tuples as random variables

probabilistic tables + database queries
→ distribution over possible worlds

```
select
from bornIn x, cityIn y
where x.city=y.city
```

one world

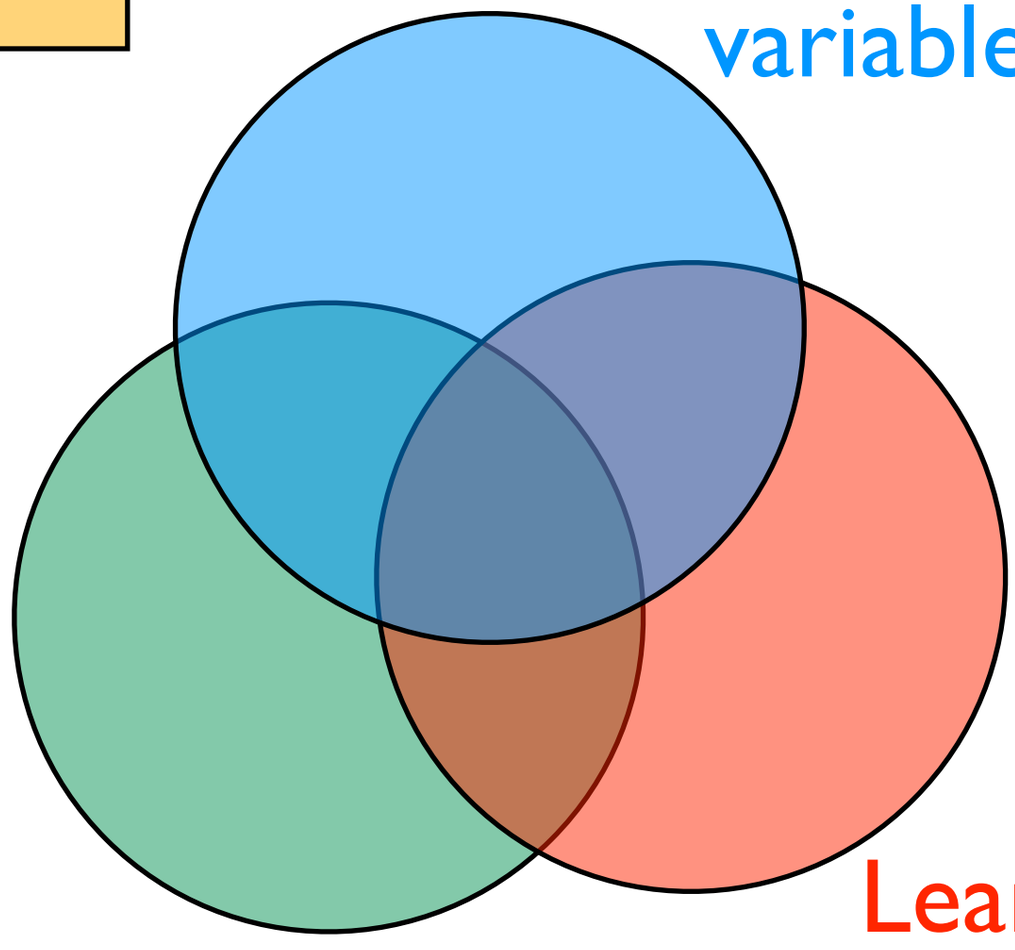
bornIn

person	city
ann	london
bob	york
eve	new york
tom	paris

cityIn

city	country
london	uk
york	uk
paris	usa

relational database



Learning

Example: Information Extraction

Recently-Learned Facts twitter Refresh

instance	iteration	date learned	confidence
kelly_andrews is a female	826	29-mar-2014	98.7
investment_next_year is an economic sector	829	10-apr-2014	95.3
shibenik is a geopolitical entity that is an organization	829	10-apr-2014	97.2
quality_web_design_work is a character trait	826	29-mar-2014	91.0
mercedes_benz_cls_by_carlsson is an automobile manufacturer	829	10-apr-2014	95.2
social_work is an academic program at the university rutgers_university	827	02-apr-2014	93.8
dante wrote the book the_divine_comedy	826	29-mar-2014	93.8
willie_aames was born in the city los_angeles	831	16-apr-2014	100.0
kitt_peak is a mountain in the state or province arizona	831	16-apr-2014	96.9
greenwich is a park in the city london	831	16-apr-2014	100.0

instances for many different relations

degree of certainty

NELL: http://rtw.ml.cmu.edu/rtw/

Querying: probabilistic db

ProducesProduct

Company	Product	P
sony	walkman	0.96
microsoft	mac_os_x	0.96
ibm	personal_computer	0.96
microsoft	mac_os	0.9
adobe	adobe_indesign	0.9
adobe	adobe_dreamweaver	0.87
...	...	...

HeadquarteredIn

Company	City	P
microsoft	redmond	1.00
ibm	san_jose	0.99
emirates_airlines	dubai	0.93
honda	torrance	0.93
horizon	seattle	0.93
egyptair	cairo	0.93
adobe	san_jose	0.93
...	...	...

[Example from Suciu et al 2011]

Querying: probabilistic db

ProducesProduct

Company	Product	P
sony	walkman	0.96
microsoft	mac_os_x	0.96
ibm	personal_computer	0.96
microsoft	mac_os	0.9
adobe	adobe_indesign	0.9
adobe	adobe_dreamweaver	0.87
...	...	...

HeadquarteredIn

Company	City	P
microsoft	redmond	1.00
ibm	san_jose	0.99
emirates_airlines	dubai	0.93
honda	torrance	0.93
horizon	seattle	0.93
egyptair	cairo	0.93
adobe	san_jose	0.93
...	...	...

```
select x.Product, x.Company
from ProducesProduct x, HeadquarteredIn y
where x.Company=y.Company and
y.City='san_jose'
```

same query -

probabilities handled implicitly

Product	Company	P
personal_computer	ibm	0.95
adobe_indesign	adobe	0.83
adobe_dreamweaver	adobe	0.80

[Example from Suciu et al 2011]

Querying: probabilistic db

ProducesProduct

Company	Product	P
sony	walkman	0.96
microsoft	mac_os_x	0.96
ibm	personal_computer	0.96
microsoft	mac_os	0.9
adobe	adobe_indesign	0.9
adobe	adobe_dreamweaver	0.87
...	...	...

HeadquarteredIn

Company	City	P
microsoft	redmond	1.00
ibm	san_jose	0.99
emirates_airlines	dubai	0.93
honda	torrance	0.93
horizon	seattle	0.93
egyptair	cairo	0.93
adobe	san_jose	0.93
...	...	...

```
select x.Product, x.Company
from ProducesProduct x, HeadquarteredIn y
where x.Company=y.Company and
y.City='san_jose'
```

0.96x0.99=0.95

Product	Company	P
personal_computer	ibm	0.95
adobe_indesign	adobe	0.83
adobe_dreamweaver	adobe	0.80

[Example from Suciu et al 2011]

Querying: probabilistic db

ProducesProduct

Company	Product	P
sony	walkman	0.96
microsoft	mac_os_x	0.96
ibm	personal_computer	0.96
microsoft	mac_os	0.9
adobe	adobe_indesign	0.9
adobe	adobe_dreamweaver	0.87
...	...	...

HeadquarteredIn

Company	City	P
microsoft	redmond	1.00
ibm	san_jose	0.99
emirates_airlines	dubai	0.93
honda	torrance	0.93
horizon	seattle	0.93
egyptair	cairo	0.93
adobe	san_jose	0.93
...	...	...

```
select x.Product, x.Company
from ProducesProduct x, HeadquarteredIn y
where x.Company=y.Company and
y.City='san_jose'
```

0.9x0.93=0.83

Product	Company	P
personal_computer	ibm	0.95
adobe_indesign	adobe	0.83
adobe_dreamweaver	adobe	0.80

[Example from Suciu et al 2011]

Querying: probabilistic db

ProducesProduct

Company	Product	P
sony	walkman	0.96
microsoft	mac_os_x	0.96
ibm	personal_computer	0.96
microsoft	mac_os	0.9
adobe	adobe_indesign	0.9
adobe	adobe_dreamweaver	0.87
...	...	...

HeadquarteredIn

Company	City	P
microsoft	redmond	1.00
ibm	san_jose	0.99
emirates_airlines	dubai	0.93
honda	torrance	0.93
horizon	seattle	0.93
egyptair	cairo	0.93
adobe	san_jose	0.93
...	...	...

```
select x.Product, x.Company
from ProducesProduct x, HeadquarteredIn y
where x.Company=y.Company and
y.City='san_jose'
```

Product	Company	P
personal_computer	ibm	0.95
adobe_indesign	adobe	0.83
adobe_dreamweaver	adobe	0.80

0.87x0.93=0.80

[Example from Suciu et al 2011]

Querying: probabilistic db

ProducesProduct

Company	Product	P
sony	walkman	0.96
microsoft	mac_os_x	0.96
ibm	personal_computer	0.96
microsoft	mac_os	0.9
adobe	adobe_indesign	0.9
adobe	adobe_dreamweaver	0.87
...	...	...

HeadquarteredIn

Company	City	P
microsoft	redmond	1.00
ibm	san_jose	0.99
emirates_airlines	dubai	0.93
honda	torrance	0.93
horizon	seattle	0.93
egyptair	cairo	0.93
adobe	san_jose	0.93
...	...	...

```
select x.Product, x.Company
from ProducesProduct x, HeadquarteredIn y
where x.Company=y.Company and
y.City='san_jose'
```

answer tuples ranked by probability

Product	Company	P
personal_computer	ibm	0.95
adobe_indesign	adobe	0.83
adobe_dreamweaver	adobe	0.80

[Example from Suciu et al 2011]

PDB with tuple-level uncertainty in ProbLog?

ProducesProduct

Company	Product	P
sony	walkman	0.96
microsoft	mac_os_x	0.96
ibm	personal_computer	0.96
microsoft	mac_os	0.9
adobe	adobe_indesign	0.9
adobe	adobe_dreamweaver	0.87
...	...	...

[Example from Suciu et al 2011]

PDB with tuple-level uncertainty in ProbLog?

ProducesProduct

Company	Product	P
sony	walkman	0.96
microsoft	mac_os_x	0.96
ibm	personal_computer	0.96
microsoft	mac_os	0.9
adobe	adobe_indesign	0.9
adobe	adobe_dreamweaver	0.87
...	...	...

```
0.96::producesProduct(sony,walkman).
0.96::producesProduct(microsoft,mac_os_x).
0.96::producesProduct(ibm,personal_computer).
0.9::producesProduct(microsoft,mac_os).
0.9::producesProduct(adobe,adobe_indesign).
0.87::producesProduct(adobe,adobe_dreamweaver).
...
```


[Example from Suciu et al 2011]

PDB with tuple-level uncertainty in ProbLog?

```
select x.Product, x.Company
from ProducesProduct x, HeadquarteredIn y
where x.Company=y.Company and y.City='san_jose'
```

PDB with tuple-level uncertainty in ProbLog?

```
select x.Product, x.Company
from ProducesProduct x, HeadquarteredIn y
where x.Company=y.Company and y.City='san_jose'
```

```
result(Product,Company) :-
    producesProduct(Company,Product),
    headquarteredIn(Company,san_jose).
query(result(_,_)).
```

PDB with tuple-level uncertainty in ProbLog?

```
select x.Product, x.Company
from ProducesProduct x, HeadquarteredIn y
where x.Company=y.Company and y.City='san_jose'
```

```
result(Product,Company) :-
    producesProduct(Company,Product),
    headquarteredIn(Company,san_jose).
query(result(_,_)).
```

PDB with attribute-level uncertainty in ProbLog?

color

item	color	P
mug	green blue	0.65 0.35
plate	pink red purple	0.23 0.14 0.63

PDB with attribute-level uncertainty in ProbLog?

color

item	color	P
mug	green blue	0.65 0.35
plate	pink red purple	0.23 0.14 0.63

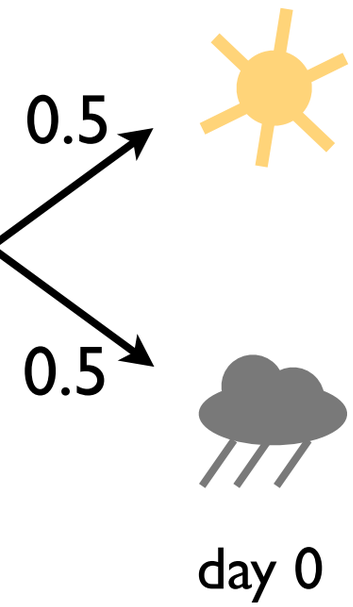
```
0.65::color(mug,green); 0.35::color(mug,blue).
0.23::color(plate,pink); 0.14::color(plate,red);
                                  0.63::color(plate,purple).
```

ProbLog by example:

Rain or sun?

ProbLog by example:

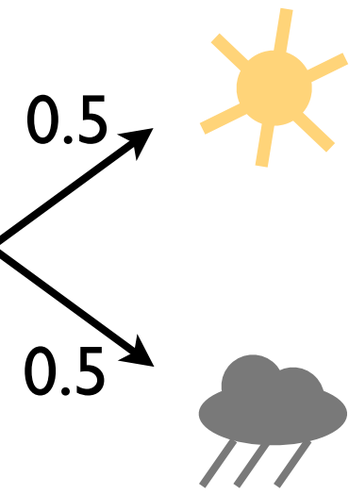
Rain or sun?



day 0

ProbLog by example:

Rain or sun?

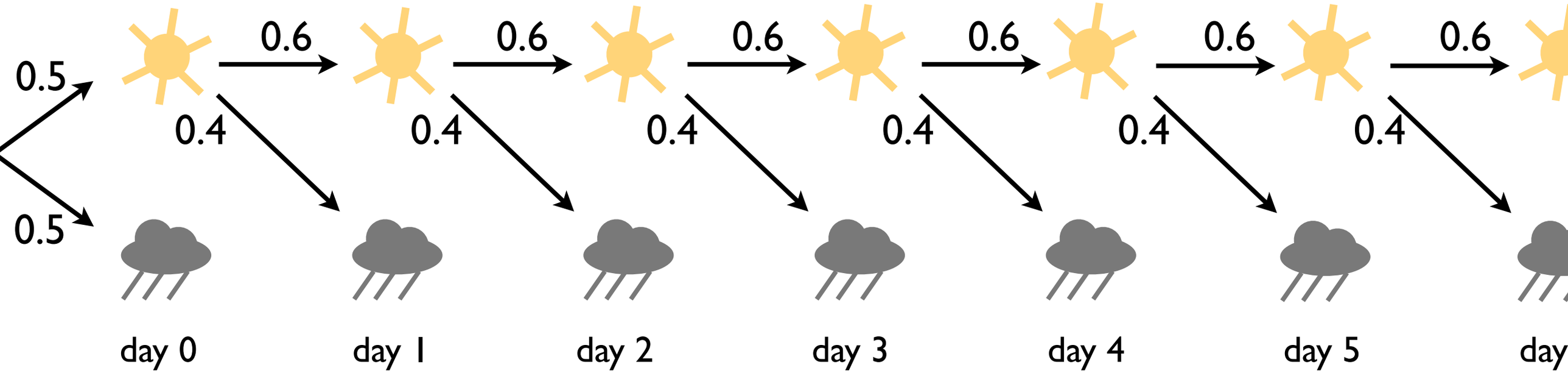


day 0

```
0.5::weather(sun,0) ; 0.5::weather(rain,0).
```

ProbLog by example:

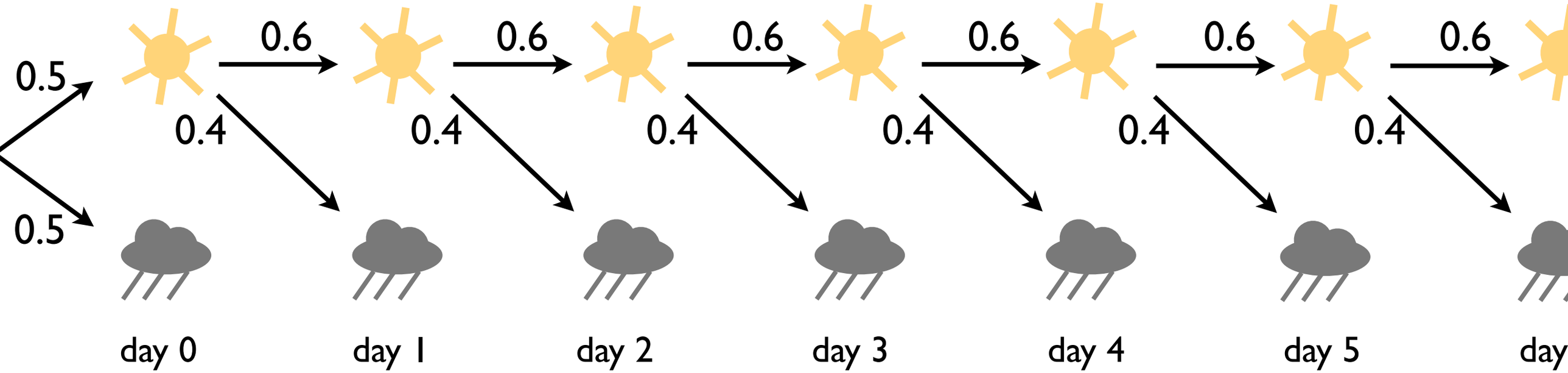
Rain or sun?




```
0.5::weather(sun,0) ; 0.5::weather(rain,0).
```

ProbLog by example:

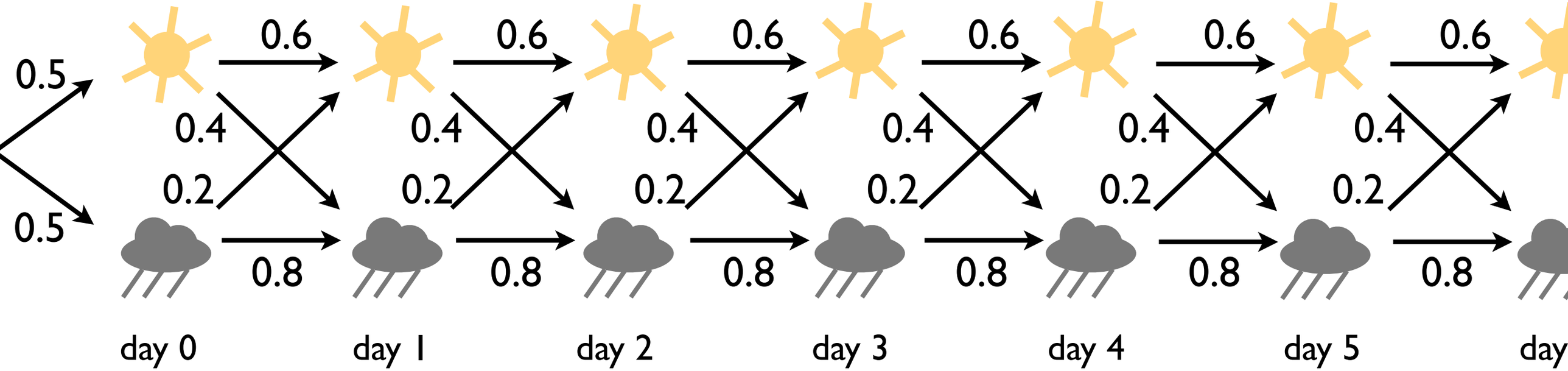
Rain or sun?




```
0.5::weather(sun,0) ; 0.5::weather(rain,0).
```

ProbLog by example:

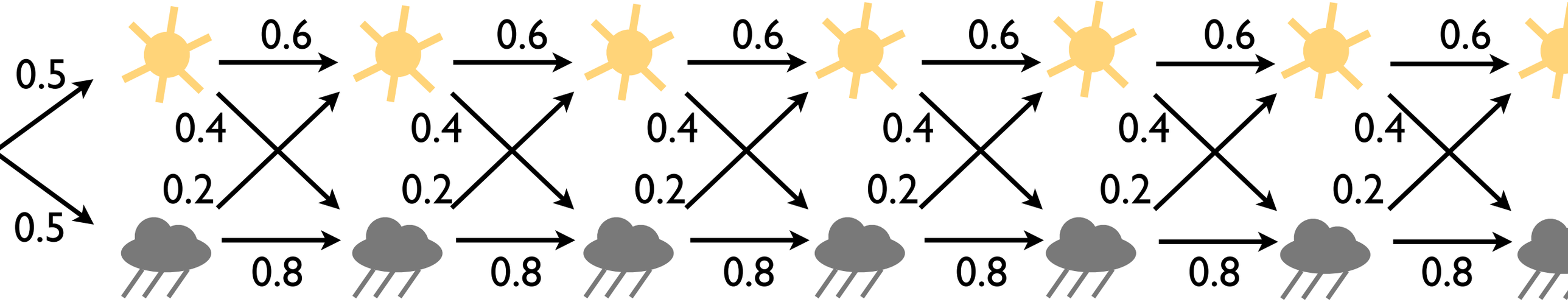
Rain or sun?




```
0.5::weather(sun,0) ; 0.5::weather(rain,0).
```

ProbLog by example:

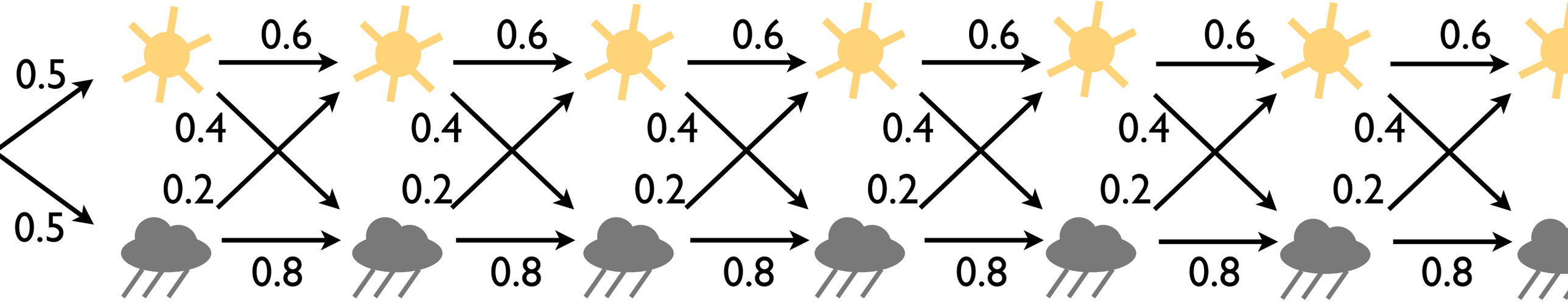
Rain or sun?




```
0.5::weather(sun,0) ; 0.5::weather(rain,0).

0.6::weather(sun,T) ; 0.4::weather(rain,T)
                    :- T>0, Tprev is T-1, weather(sun,Tprev).
```

ProbLog by example:

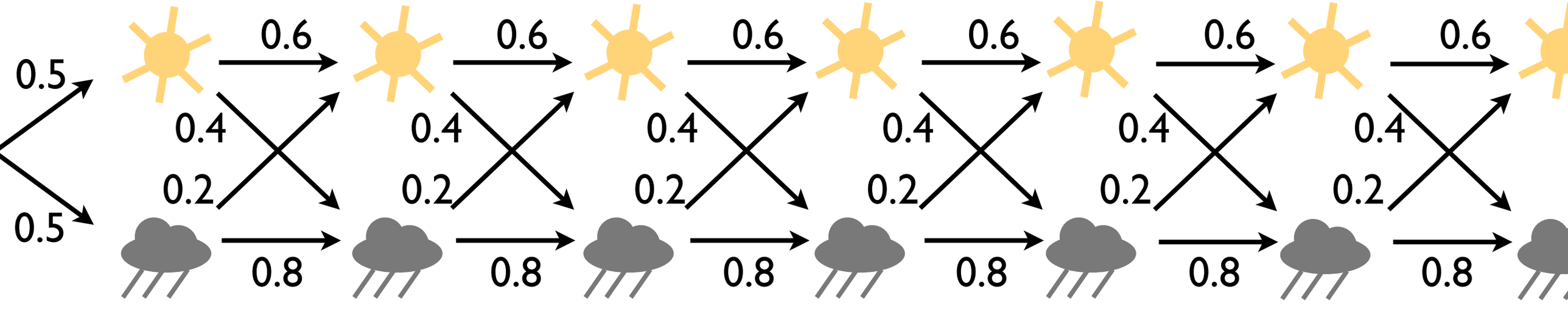
Rain or sun?




```
0.5::weather(sun,0) ; 0.5::weather(rain,0).

0.6::weather(sun,T) ; 0.4::weather(rain,T)
                    :- T>0, Tprev is T-1, weather(sun,Tprev).
0.2::weather(sun,T) ; 0.8::weather(rain,T)
                    :- T>0, Tprev is T-1, weather(rain,Tprev).
```

ProbLog by example:

Rain or sun?



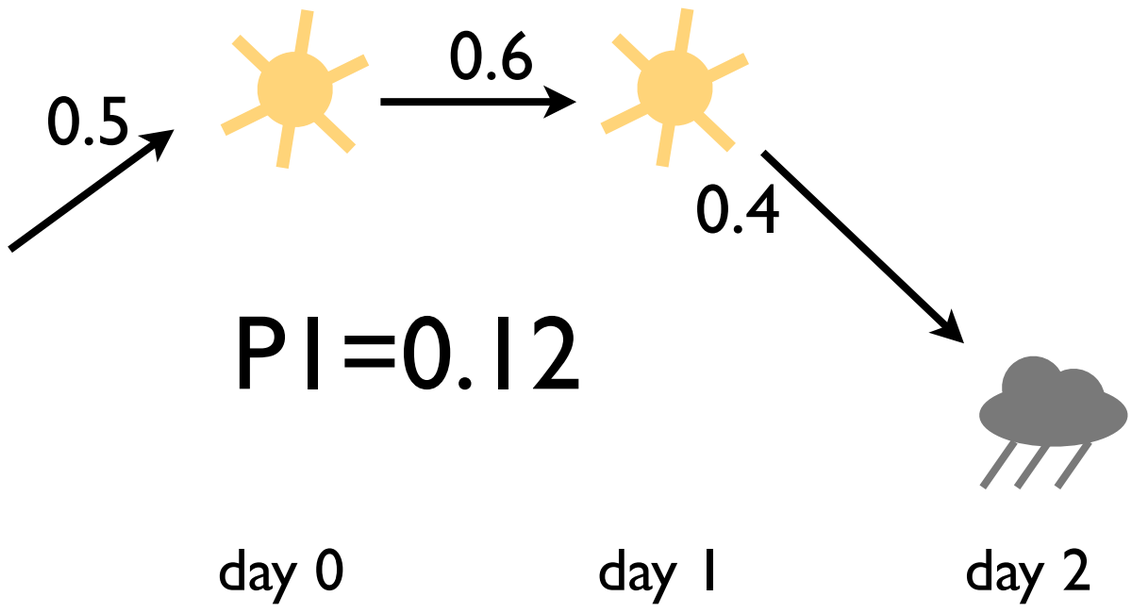

```
0.5::weather(sun,0) ; 0.5::weather(rain,0).

0.6::weather(sun,T) ; 0.4::weather(rain,T)
                    :- T>0, Tprev is T-1, weather(sun,Tprev).
0.2::weather(sun,T) ; 0.8::weather(rain,T)
                    :- T>0, Tprev is T-1, weather(rain,Tprev).
```

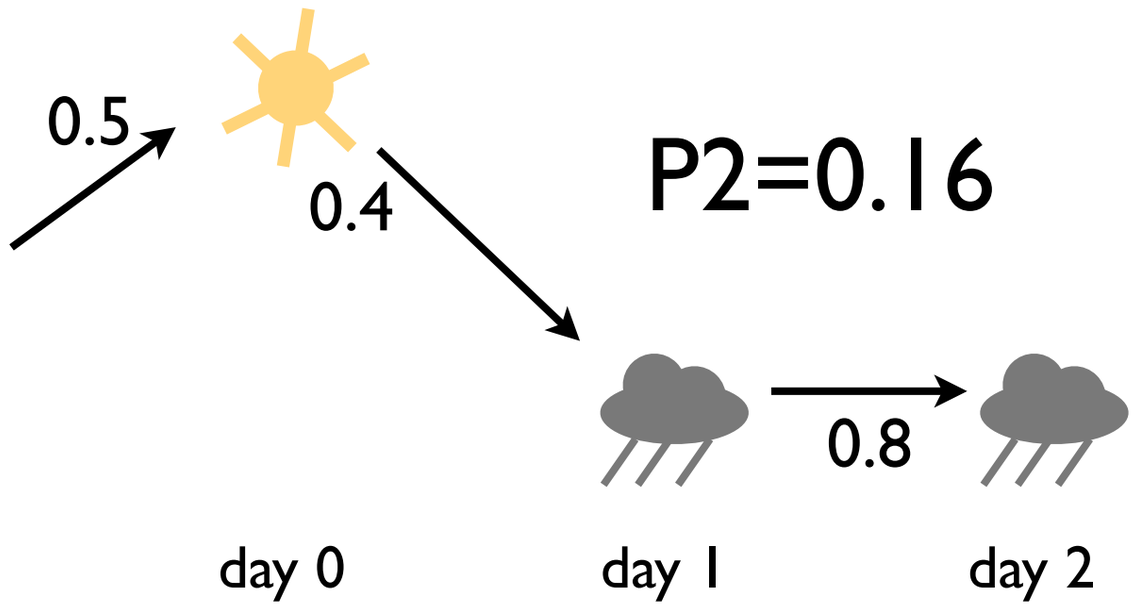
infinite possible worlds! BUT: finitely many partial worlds suffice to answer any given ground query

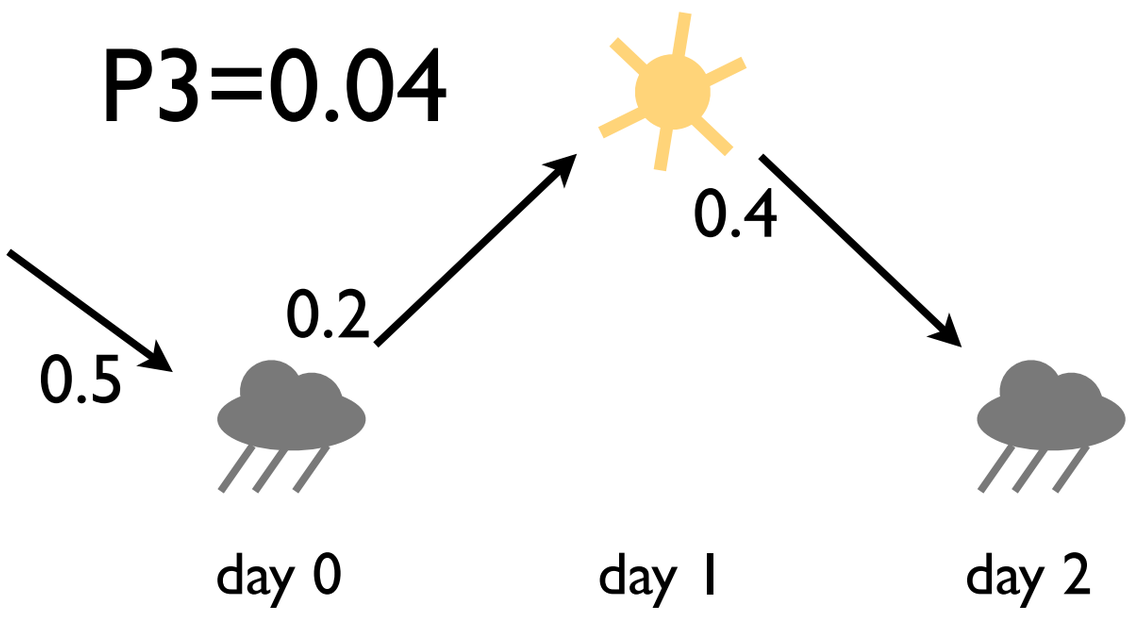
Possible worlds

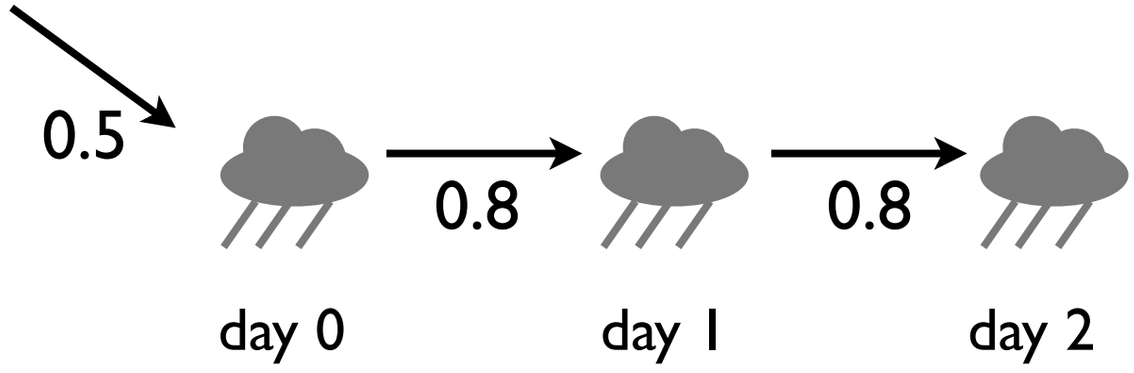
```
?- weather(rain,2).
```

P=P1+P2+P3+P4



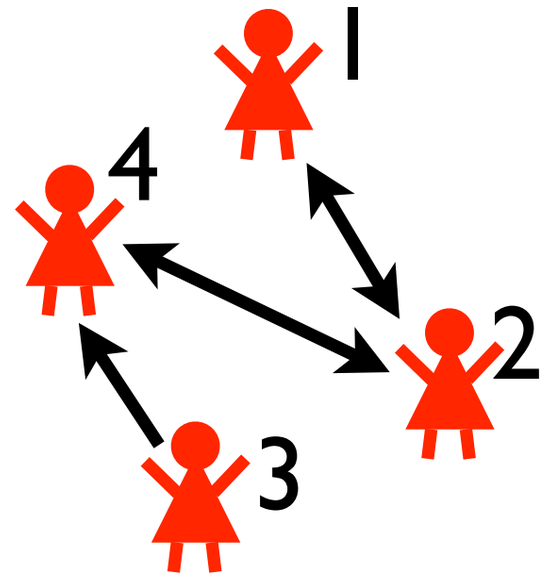






ProbLog by example:

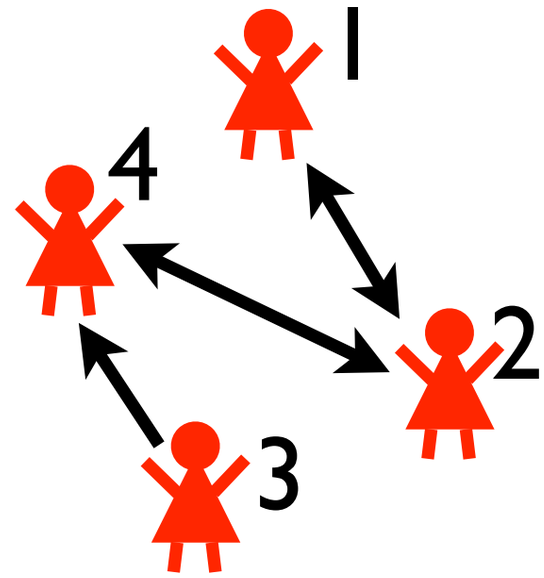
Friends & smokers




```
person(1).
person(2).
person(3).
person(4).

friend(1,2).
friend(2,1).
friend(2,4).
friend(3,4).
friend(4,2).
```

ProbLog by example:

Friends & smokers



typed probabilistic facts

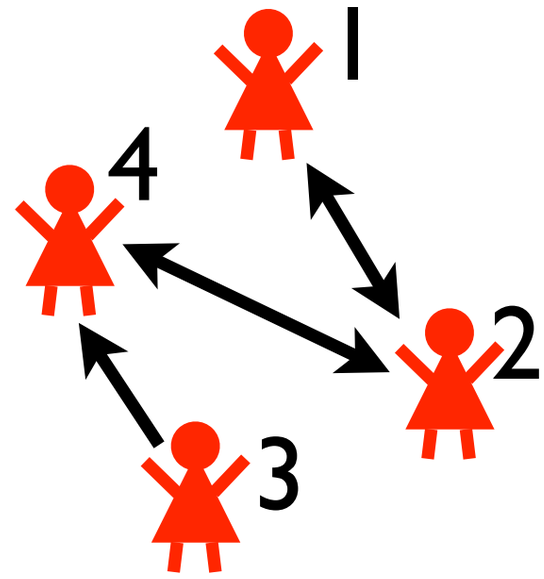
= a probabilistic fact for each grounding

```
0.3::stress(X):- person(X).
0.2::influences(X,Y):-
                person(X), person(Y).
```

```
person(1).
person(2).
person(3).
person(4).

friend(1,2).
friend(2,1).
friend(2,4).
friend(3,4).
friend(4,2).
```

ProbLog by example:

Friends & smokers



typed probabilistic facts

= a probabilistic fact for each grounding

```
0.3::stress(X):- person(X).
0.2::influences(X,Y):-
              person(X), person(Y).
```

```
person(1).
person(2).
person(3).
person(4).
```

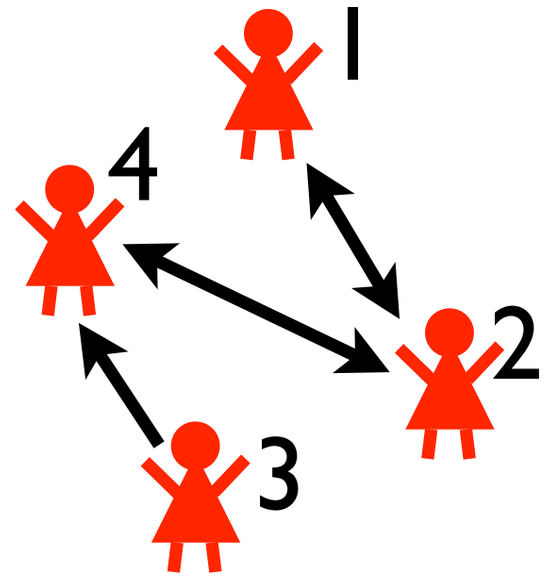
```
0.3::stress(1).
0.3::stress(2).
0.3::stress(3).
0.3::stress(4).
```

```
0.2::influences(1,1).
0.2::influences(1,2).
0.2::influences(1,3).
0.2::influences(1,4).
0.2::influences(2,1).
...
0.2::influences(4,2).
0.2::influences(4,3).
0.2::influences(4,4).
```

```
friend(1,2).
friend(2,1).
friend(2,4).
friend(3,4).
friend(4,2).
```

ProbLog by example:

Friends & smokers

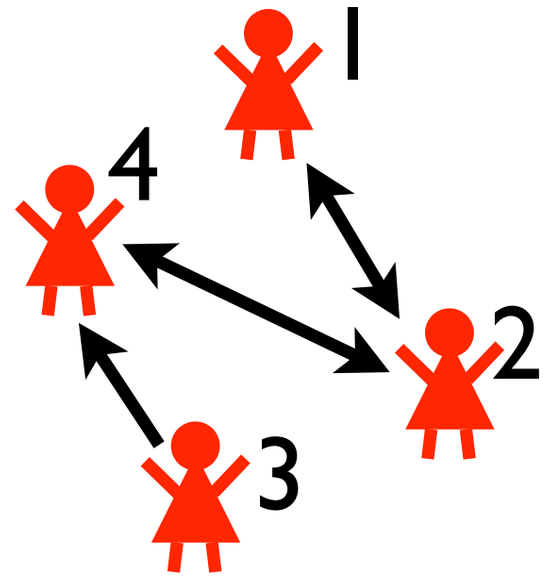



```
0.3::stress(X):- person(X).
0.2::influences(X,Y):-
                person(X), person(Y).
```

```
person(1).
person(2).
person(3).
person(4).

friend(1,2).
friend(2,1).
friend(2,4).
friend(3,4).
friend(4,2).
```

ProbLog by example:

Friends & smokers

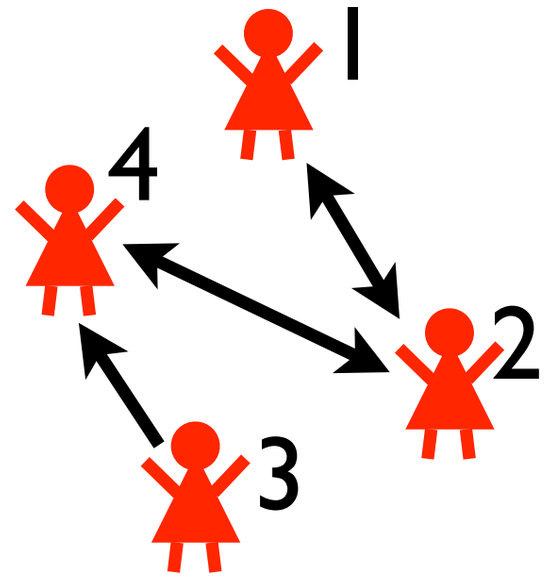



```
0.3::stress(X):- person(X).
0.2::influences(X,Y):-
              person(X), person(Y).

smokes(X) :- stress(X).
smokes(X) :-
     friend(X,Y), influences(Y,X), smokes(Y).
```

```
person(1).
person(2).
person(3).
person(4).

friend(1,2).
friend(2,1).
friend(2,4).
friend(3,4).
friend(4,2).
```

ProbLog by example:

Friends & smokers

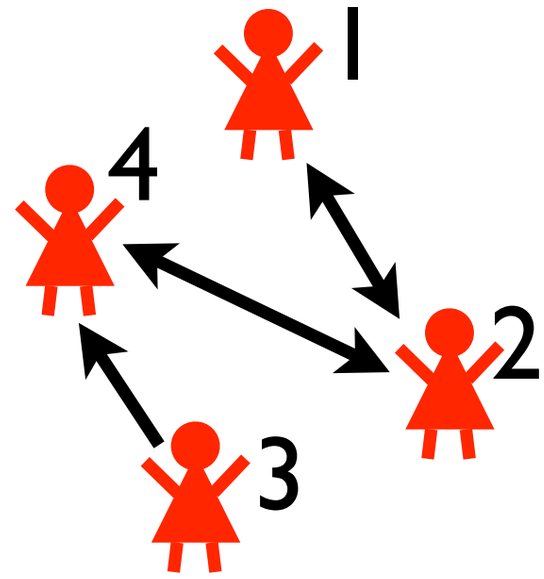



```
0.3::stress(X):- person(X).
0.2::influences(X,Y):-
              person(X), person(Y).

smokes(X) :- stress(X).
smokes(X) :-
     friend(X,Y), influences(Y,X), smokes(Y).
```

```
person(1).
person(2).
person(3).
person(4).

friend(1,2).
friend(2,1).
friend(2,4).
friend(3,4).
friend(4,2).
```

ProbLog by example:

Friends & smokers




```
0.3::stress(X):- person(X).
0.2::influences(X,Y):-
                person(X), person(Y).

smokes(X)  :- stress(X).
smokes(X)  :-
     friend(X,Y), influences(Y,X), smokes(Y).

0.4::asthma(X) <- smokes(X).
```

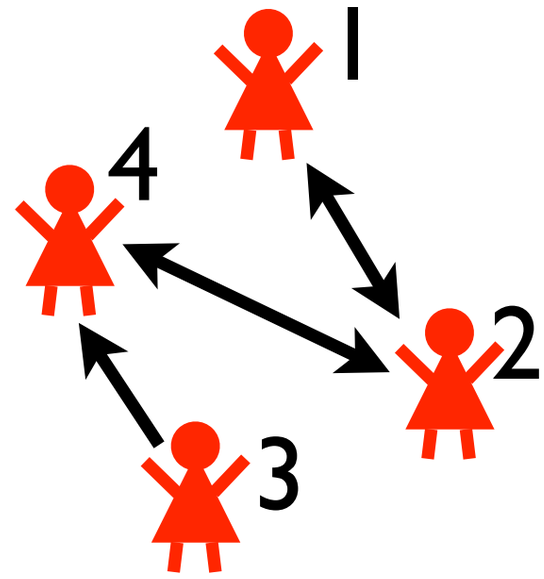
```
person(1).
person(2).
person(3).
person(4).

friend(1,2).
friend(2,1).
friend(2,4).
friend(3,4).
friend(4,2).
```

annotated disjunction with implicit head atom:

with probability 0.6, nothing happens

ProbLog by example:

Friends & smokers




```
0.3::stress(X):- person(X).
0.2::influences(X,Y):-
              person(X), person(Y).

smokes(X) :- stress(X).
smokes(X) :-
     friend(X,Y), influences(Y,X), smokes(Y).

0.4::asthma(X) <- smokes(X).
```

```
person(1).
person(2).
person(3).
person(4).

friend(1,2).
friend(2,1).
friend(2,4).
friend(3,4).
friend(4,2).
```

Concept: flexible probability

ProbLog by example:

Limited Luggage

```
weight(skis,6).
weight(boots,4).
weight(helmet,3).
weight(gloves,2).
```

Concept: flexible probability

ProbLog by example:

Limited Luggage

```
weight(skis,6).
weight(boots,4).
weight(helmet,3).
weight(gloves,2).
P::pack(Item)  - weight(Item,Weight),  P is 1.0/Weight.
```

flexible probability:

computed from the weight of the item

Concept: flexible probability

ProbLog by example:

Limited Luggage

```
1/6::pack(skis).
1/4::pack(boots).
1/3::pack(helmet).
1/2::pack(gloves).
```

```
weight(skis,6).
weight(boots,4).
weight(helmet,3).
weight(gloves,2).
```

```
P::pack(Item)   - weight(Item,Weight),   P is 1.0/Weight.
```

flexible probability:

computed from the weight of the item

Concept: flexible probability

ProbLog by example:

Limited Luggage

```
weight(skis,6).
weight(boots,4).
weight(helmet,3).
weight(gloves,2).
P::pack(Item) :- weight(Item,Weight),  P is 1.0/Weight.
excess(Limit) :- excess([skis,boots,helmet,gloves],Limit).
```

list of all items

Concept: flexible probability

ProbLog by example:

Limited Luggage

```
weight(skis,6).
weight(boots,4).
weight(helmet,3).
weight(gloves,2).
P::pack(Item) :- weight(Item,Weight),  P is 1.0/Weight.
excess(Limit) :- excess([skis,boots,helmet,gloves],Limit).

excess([],Limit) :- Limit<0.
excess([I|R],Limit) :-
   pack(I), weight(I,W), L is Limit-W, excess(R,L).
excess([I|R],Limit) :-
   \+pack(I), excess(R,Limit).
```

Concept: flexible probability

ProbLog by example:

Limited Luggage

```
weight(skis,6).
weight(boots,4).
weight(helmet,3).
weight(gloves,2).
P::pack(Item) :- weight(Item,Weight),  P is 1.0/Weight.
excess(Limit) :- excess([skis,boots,helmet,gloves],Limit).

excess([],Limit) :- Limit<0.
excess([I|R],Limit) :-
   pack(I), weight(I,W), L is Limit-W, excess(R,L).
excess([I|R],Limit) :-
   \+pack(I), excess(R,Limit).
```

pack first item, decrease limit by its weight, and continue with rest of items

Concept: flexible probability

ProbLog by example:

Limited Luggage

```
weight(skis,6).
weight(boots,4).
weight(helmet,3).
weight(gloves,2).
P::pack(Item) :- weight(Item,Weight),  P is 1.0/Weight.
excess(Limit) :- excess([skis,boots,helmet,gloves],Limit).

excess([],Limit) :- Limit<0.
excess([I|R],Limit) :-
   pack(I), weight(I,W), L is Limit-W, excess(R,L).
excess([I|R],Limit) :-
   \+pack(I), excess(R,Limit).
```

do **not** pack first item,
continue with rest of items

Concept: flexible probability

ProbLog by example:

Limited Luggage

```
weight(skis,6).
weight(boots,4).
weight(helmet,3).
weight(gloves,2).
P::pack(Item) :- weight(Item,Weight),  P is 1.0/Weight.
excess(Limit) :- excess([skis,boots,helmet,gloves],Limit).

excess([],Limit) :- Limit<0.
excess([I|R],Limit) :
   pack(I), weight(I,W), L is Limit-W, excess(R,L).
excess([I|R],Limit) :-
   \+pack(I), excess(R,Limit).
```

no items left: did we add too much?

Concept: flexible probability

ProbLog by example:

Limited Luggage

```
weight(skis,6).
weight(boots,4).
weight(helmet,3).
weight(gloves,2).
P::pack(Item) :- weight(Item,Weight),  P is 1.0/Weight.
excess(Limit) :- excess([skis,boots,helmet,gloves],Limit).

excess([],Limit) :- Limit<0.
excess([I|R],Limit) :-
   pack(I), weight(I,W), L is Limit-W, excess(R,L).
excess([I|R],Limit) :-
   \+pack(I), excess(R,Limit).
```

ProbLog

- **probabilistic choices** + their **consequences**
- probability distribution over **possible worlds**
- how to efficiently answer **questions**?
  - most probable world (MPE inference)
  - probability of query (computing marginals)

Summary: ProbLog Syntax

- input database: ground facts

```
person(bob).
```

- probabilistic facts

```
0.5::stress(bob).
```

- annotated disjunctions

```
0.5::stress(X) :- person(X).
0.4::a(X); 0.3::b(X); 0.2::c(X); 0.1::d(X) :- q(X).
0.5::weather(sun,0) ; 0.5::weather(rain,0).
```

- flexible probabilities

```
P::pack(Item) :- weight(Item,W), P is 1.0/W.
```

- Prolog clauses

```
smokes(X) :- influences(Y,X), smokes(Y).
excess([I|R],Limit) :- \+pack(I), excess(R,Limit).
```

Mutually Exclusive Rules:

no two rules apply simultaneously

```
0.5::weather(sun,0) ; 0.5::weather(rain,0).

0.6::weather(sun,T) ; 0.4::weather(rain,T)
                :- T>0, Tprev is T-1, weather(sun,Tprev).

0.2::weather(sun,T) ; 0.8::weather(rain,T)
                :- T>0, Tprev is T-1, weather(rain,Tprev).
```

Mutually Exclusive Rules:

no two rules apply simultaneously

first rule for day 0, others for later days

```
0.5::weather(sun,0) ; 0.5::weather(rain,0).

0.6::weather(sun,T) ; 0.4::weather(rain,T)
                :- T>0, Tprev is T-1, weather(sun,Tprev).

0.2::weather(sun,T) ; 0.8::weather(rain,T)
                :- T>0, Tprev is T-1, weather(rain,Tprev).
```

Mutually Exclusive Rules:

no two rules apply simultaneously

first rule for day 0, others for later days

day 0: either sun or rain

```
0.5::weather(sun,0) ; 0.5::weather(rain,0).

0.6::weather(sun,T) ; 0.4::weather(rain,T)
                :- T>0, Tprev is T-1, weather(sun,Tprev).

0.2::weather(sun,T) ; 0.8::weather(rain,T)
                :- T>0, Tprev is T-1, weather(rain,Tprev).
```

Mutually Exclusive Rules:

no two rules apply simultaneously

first rule for day 0, others for later days

day 0: either sun or rain

```
0.5::weather(sun,0) ; 0.5::weather(rain,0).

0.6::weather(sun,T) ; 0.4::weather(rain,T)
                    :- T>0, Tprev is T-1, weather(sun,Tprev).

0.2::weather(sun,T) ; 0.8::weather(rain,T)
                    :- T>0, Tprev is T-1, weather(rain,Tprev).
```

rules for T>0 cover mutually exclusive cases on previous day

[Sato and Kameya, JAIR 01]

PRISM

- Another probabilistic Prolog based on the distribution semantics
- Mutual exclusiveness assumption
  - allows for efficient inference by dynamic programming, cf. probabilistic grammars
  - but excludes certain models, e.g., smokers

PRISM

- "multi-valued random switches" = annotated disjunctions with body *true*
- switch gives fresh result on each call stochastic memoization
- Prolog rules
- limited support for negation (compiling away)

Weather in PRISM

```
values(init,[sun,rain]).
values(tr(_),[sun,rain]).

:- set_sw(init,[0.5,0.5]).
:- set_sw(tr(sun),[0.6,0.4]).
:- set_sw(tr(rain),[0.2,0.8]).
```

```
weather(W,Time) :-
  Time >= 0,
  msw(init,W0),
  w(0,Time,W0,W).

w(T,T,W,W).
w(Now,T,WNow,WT) :-
    Now < T,
    msw(tr(WNow),WNext),
    Next is Now+1,
    w(Next,T,WNext,WT).
```

Weather in PRISM

```
values(init,[sun,rain]).
values(tr(_),[sun,rain])
```

random variables and their values

```
:- set_sw(init,[0.5,0.5]).
:- set_sw(tr(sun),[0.6,0.4]).
:- set_sw(tr(rain),[0.2,0.8]).
```

```
weather(W,Time) :-
  Time >= 0,
  msw(init,W0),
  w(0,Time,W0,W).

w(T,T,W,W).
w(Now,T,WNow,WT) :-
    Now < T,
    msw(tr(WNow),WNext),
    Next is Now+1,
    w(Next,T,WNext,WT).
```

Weather in PRISM

```
values(init,[sun,rain]).
values(tr(_),[sun,rain])
```

random variables and their values

```
:- set_sw(init,[0.5,0.5]).
:- set_sw(tr(sun),[0.6,0.4]).
:- set_sw(tr(rain),[0.2,0.8]).
```

probability distributions

```
weather(W,Time) :-
  Time >= 0,
  msw(init,W0),
  w(0,Time,W0,W).

w(T,T,W,W).
w(Now,T,WNow,WT) :-
   Now < T,
   msw(tr(WNow),WNext),
   Next is Now+1,
   w(Next,T,WNext,WT).
```

Weather in PRISM

```
values(init,[sun,rain]).
values(tr(_),[sun,rain])
```

random variables and their values

```
:- set_sw(init,[0.5,0.5]).
:- set_sw(tr(sun),[0.6,0.4]).
:- set_sw(tr(rain),[0.2,0.8]).
```

probability distributions

```
weather(W,Time) :-
   Time >= 0,
   msw(init,W0),
   w(0,Time,W0,W).

w(T,T,W,W).
w(Now,T,WNow,WT) :-
    Now < T,
    msw(tr(WNow),WNext),
    Next is Now+1,
    w(Next,T,WNext,WT).
```

set **W0** to random value of **init**

Weather in PRISM

```
values(init,[sun,rain]).
values(tr(_),[sun,rain])
```

random variables and their values

```
:- set_sw(init,[0.5,0.5]).
:- set_sw(tr(sun),[0.6,0.4]).
:- set_sw(tr(rain),[0.2,0.8]).
```

probability distributions

```
weather(W,Time) :-
   Time >= 0,
   msw(init,W0),
   w(0,Time,W0,W).

w(T,T,W,W).
w(Now,T,WNow,WT) :-
    Now < T,
    msw(tr(WNow),WNext),
    Next is Now+1,
    w(Next,T,WNext,WT).
```

set **W0** to random value of **init**

set **WNext** to random value of **tr(WNow)**, using **fresh** value on every call

Weather in PRISM / ProbLog

```
values(init,[sun,rain]).
values(tr(_),[sun,rain]).

:- set_sw(init,[0.5,0.5]).
:- set_sw(tr(sun),[0.6,0.4]).
:- set_sw(tr(rain),[0.2,0.8]).
```

```
weather(W,Time) :-
  Time >= 0,
  msw(init,W0),
  w(0,Time,W0,W).

w(T,T,W,W).
w(Now,T,WNow,WT) :-
   Now < T,
   msw(tr(WNow),WNext),
   Next is Now+1,
   w(Next,T,WNext,WT).
```

```
0.5::init(sun) ;
  0.5::init(rain).
0.6::tr(T,sun,sun) ;
  0.4::tr(T,sun,rain).
0.2::tr(T,rain,sun) ;
  0.8::tr(T,rain,rain).
```

```
weather(W,Time) :-
  Time >= 0,
  init(W0),
  w(0,Time,W0,W).

w(T,T,W,W).
w(Now,T,WNow,WT) :-
   Now < T,
   tr(Now,WNow,WNext),
   Next is Now+1,
   w(Next,T,WNext,WT).
```

ProbLog needs to explicitly use different facts at each call

Probabilistic Prologs: Two Views

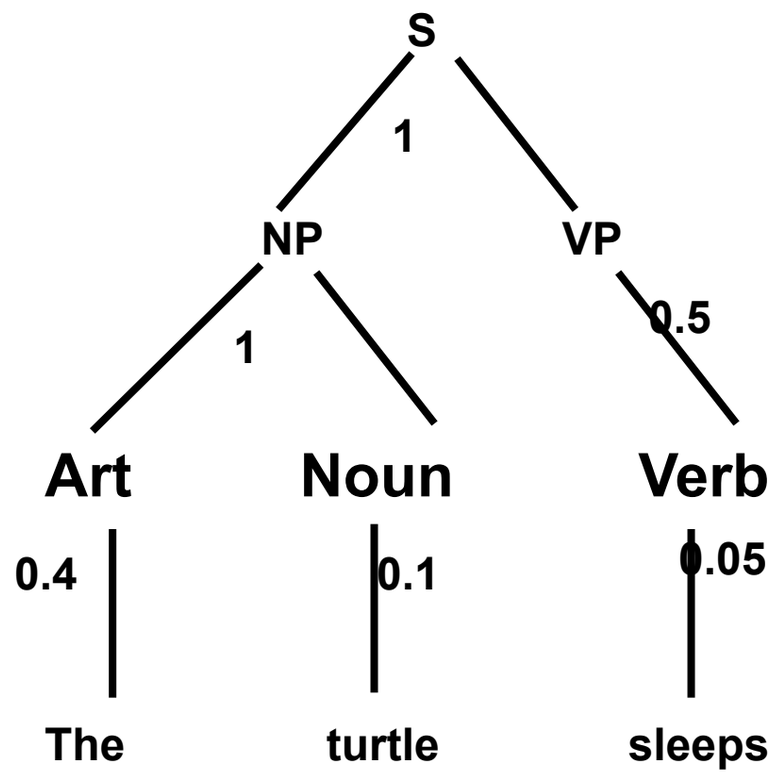
- Distribution semantics:
  - probability distribution over interpretations
  - degree of belief
- Stochastic Logic Programs (SLPs):
  - probability distribution over query answers
  - like in probabilistic grammars

Probabilistic Prologs: Two Views

- Distribution semantics:
  - probability distribution over interpretations
  - degree of belief
- Stochastic Logic Programs (SLPs):
  - probability distribution over query answers
  - like in probabilistic grammars

Probabilistic Context Free Grammars

1.0 : S -> NP, VP
1.0 : NP -> Art, Noun
0.6 : Art -> a
0.4 : Art -> the
0.1 : Noun -> turtle
0.1 : Noun -> turtles
…
0.5 : VP -> Verb
0.5 : VP -> Verb, NP
0.05 : Verb -> sleep
0.05 : Verb -> sleeps
….



P(parse tree) = 1x1x.5x.1x.4x.05

PCFGs

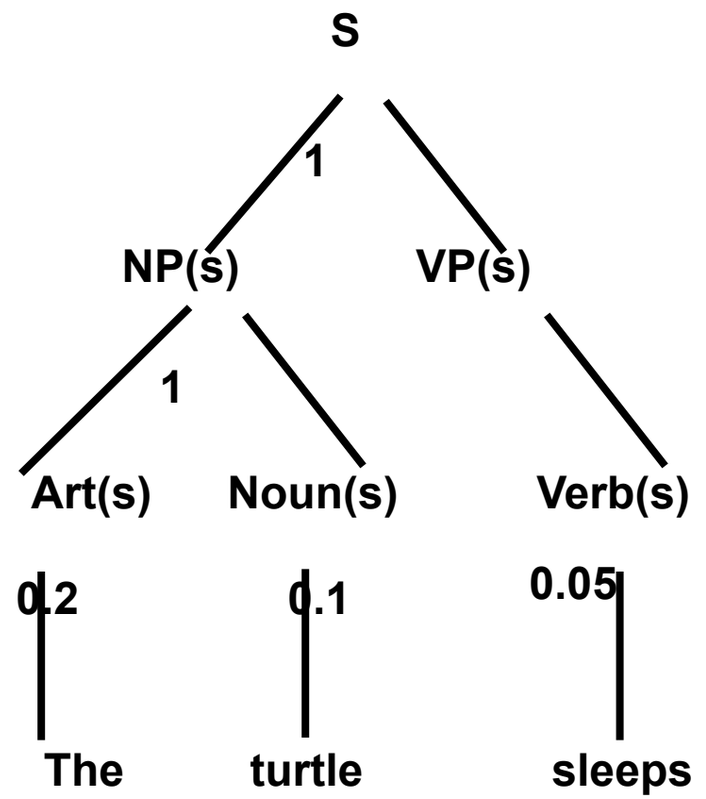
$P(parse\ tree) = \prod_i p_i^{c_i}$
where p_i is the probability of rule i
and c_i the number of times
it is used in the parse tree

$P(sentence) = \sum_{p:parsetree} P(p)$

Observe that derivations always succeed, that is
$S \to T, Q$ and $T \to R, U$
always yields
$S \to R, U, Q$

Probabilistic Definite Clause Grammar

1.0 S -> NP(Num), VP(Num)
1.0 NP(Num) -> Art(Num), Noun(Num)
0.6 Art(sing) -> a
0.2 Art(sing) -> the
0.2 Art(plur) -> the
0.1 Noun(sing) -> turtle
0.1 Noun(plur) -> turtles
…
0.5 VP(Num) -> Verb(Num)
0.5 VP(Num) -> Verb(Num), NP(Num)
0.05 Verb(sing) -> sleeps
0.05 Verb(plur) -> sleep
….



P(derivation tree) = 1x1x.5x.1x .2 x.05

Stochastic Logic Programs

[Muggleton, Cussens]

In SLP notation

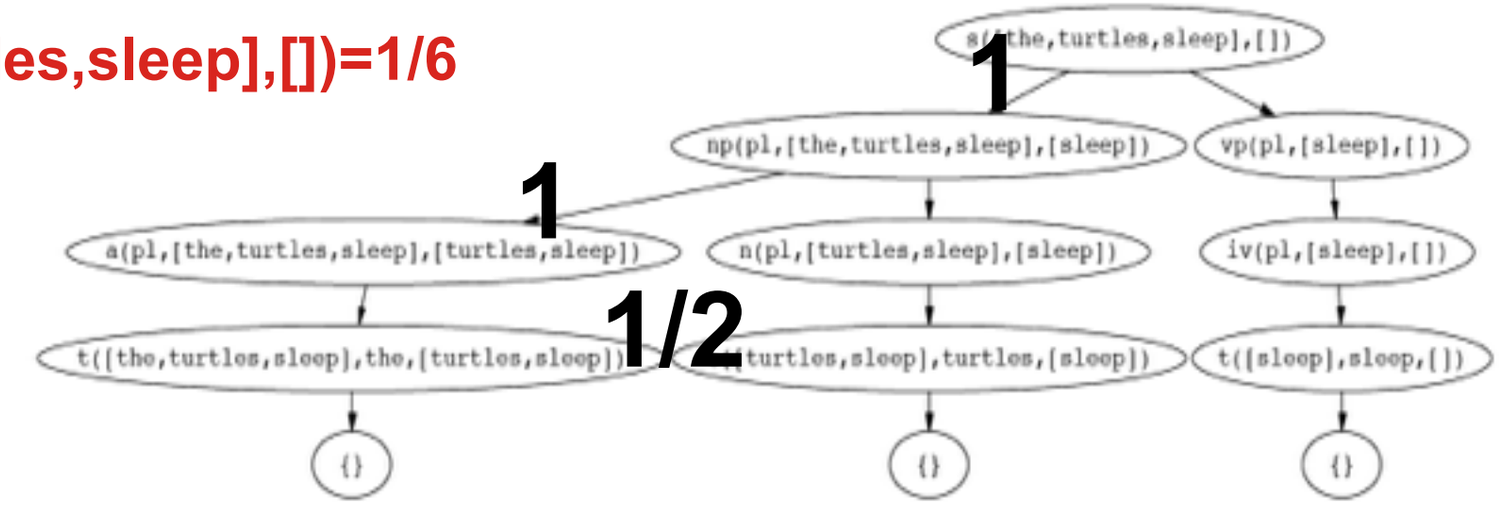
1

1/3

1/2

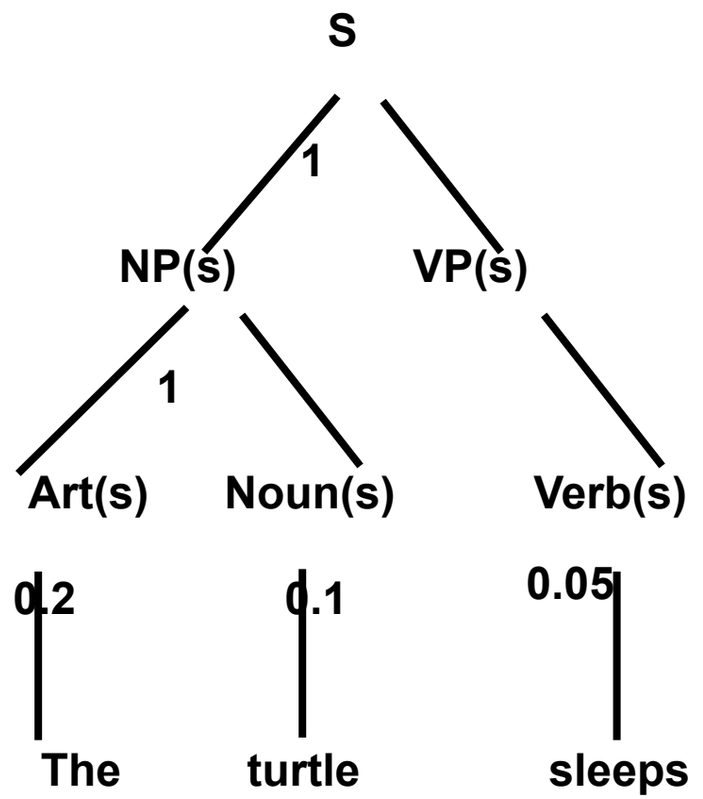
```
sentence(A, B) :- noun_phrase(C, A, D), verb_phrase(C, D, B).
noun_phrase(A, B, C) :- article(A, B, D), noun(A, D, C).
verb_phrase(A, B, C) :- intransitive_verb(A, B, C).
article(singular, A, B) :- terminal(A, a, B).
article(singular, A, B) :- terminal(A, the, B).
article(plural, A, B)   :- terminal(A, the, B).
noun(singular, A, B) :- terminal(A, turtle, B).
noun(plural, A, B)   :- terminal(A, turtles, B).
intransitive_verb(singular, A, B) :- terminal(A, sleeps, B).
intransitive_verb(plural, A, B) :- terminal(A, sleep, B).
terminal([A|B],A,B).
```

P(s([the,turtles,sleep],[])=1/6



Probabilistic DCG

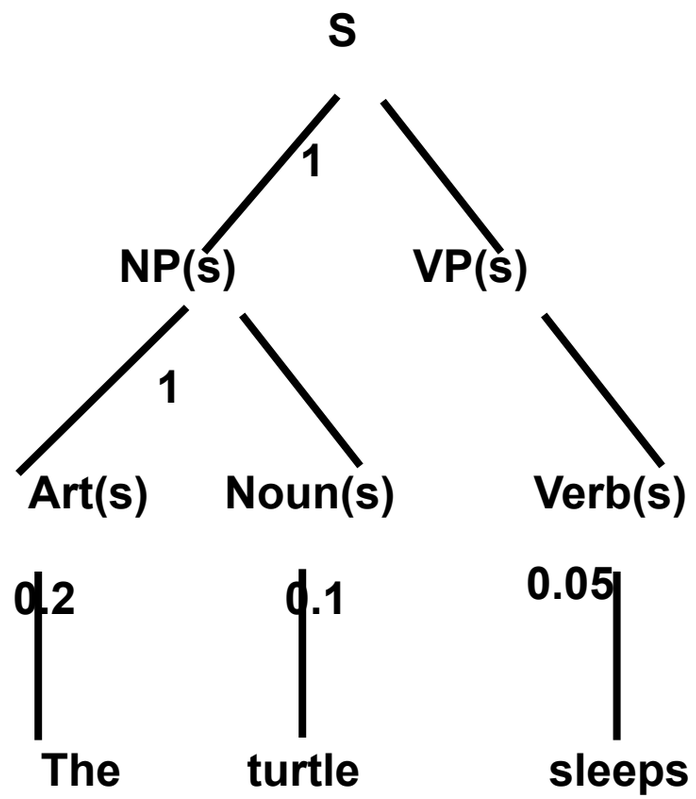
1.0 S -> NP(Num), VP(Num)
1.0 NP(Num) -> Art(Num), Noun(Num)
0.6 Art(sing) -> a
0.2 Art(sing) -> the
0.2 Art(plur) -> the
0.1 Noun(sing) -> turtle
0.1 Noun(plur) -> turtles
…
0.5 VP(Num) -> Verb(Num)
0.5 VP(Num) -> Verb(Num), NP(Num)
0.05 Verb(sing) -> sleeps
0.05 Verb(plur) -> sleep
….



P(derivation tree) = 1x1x.5x.1x .2 x.05

Probabilistic DCG

1.0 S -> NP(Num), VP(Num)
1.0 NP(Num) -> Art(Num), Noun(Num)
0.6 Art(sing) -> a
0.2 Art(sing) -> the
0.2 Art(plur) -> the
0.1 Noun(sing) -> turtle
0.1 Noun(plur) -> turtles
…
0.5 VP(Num) -> Verb(Num)
0.5 VP(Num) -> Verb(Num), NP(Num)
0.05 Verb(sing) -> sleeps
0.05 Verb(plur) -> sleep
….



P(derivation tree) = 1x1x.5x.1x .2 x.05

What about "A turtles sleeps" ?

SLPs

$P_d(derivation) = \prod_i p_i^{c_i}$
where p_i is the probability of rule i
and c_i the number of times
it is used in the parse tree

Observe that some derivations now fail due to unification,
$np(Num) \rightarrow art(Num), noun(Num)$ and $art(sing) \rightarrow a$
$noun(plural) \rightarrow turtles$

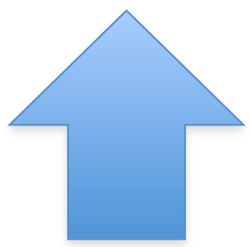
Normalization necessary
$P_s(proof) = \frac{P_d(proof)}{\sum_i P_d(proof_i)}$

ProPPR

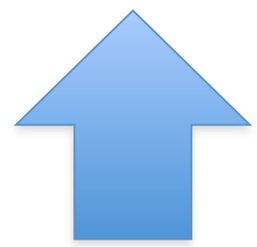
- A variation on SLPs
- Integrating concepts from Personalized Page Rank
- Fast inference and rule learning abilities
- Used by CMU group for NELL (Never Ending Learning)
- See [Wang et al. , CIKM 13, arXiv:1404.3301, Van Daele et al., ILP 14]

Sample ProPPR program….

```
about(X,Z) :- handLabeled(X,Z)          # base.
about(X,Z) :- sim(X,Y),about(Y,Z)       # prop.
sim(X,Y) :- links(X,Y)                  # sim,link.
sim(X,Y) :-
    hasWord(X,W),hasWord(Y,W),
    linkedBy(X,Y,W)                     # sim,word.
linkedBy(X,Y,W) :- true                 # by(W).
```

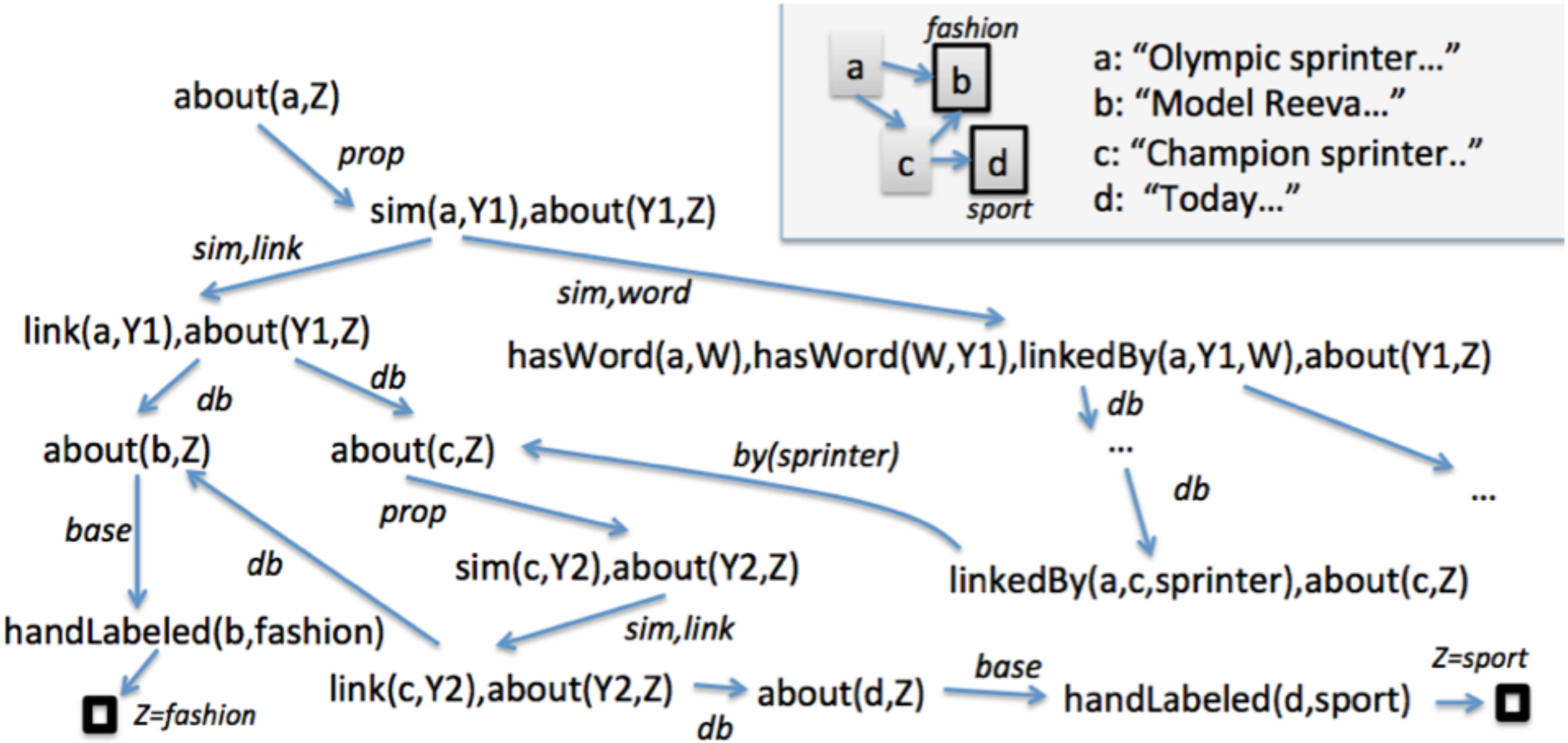


Horn rules



features of rules
(vars from head ok)

[Slide by William Cohen]

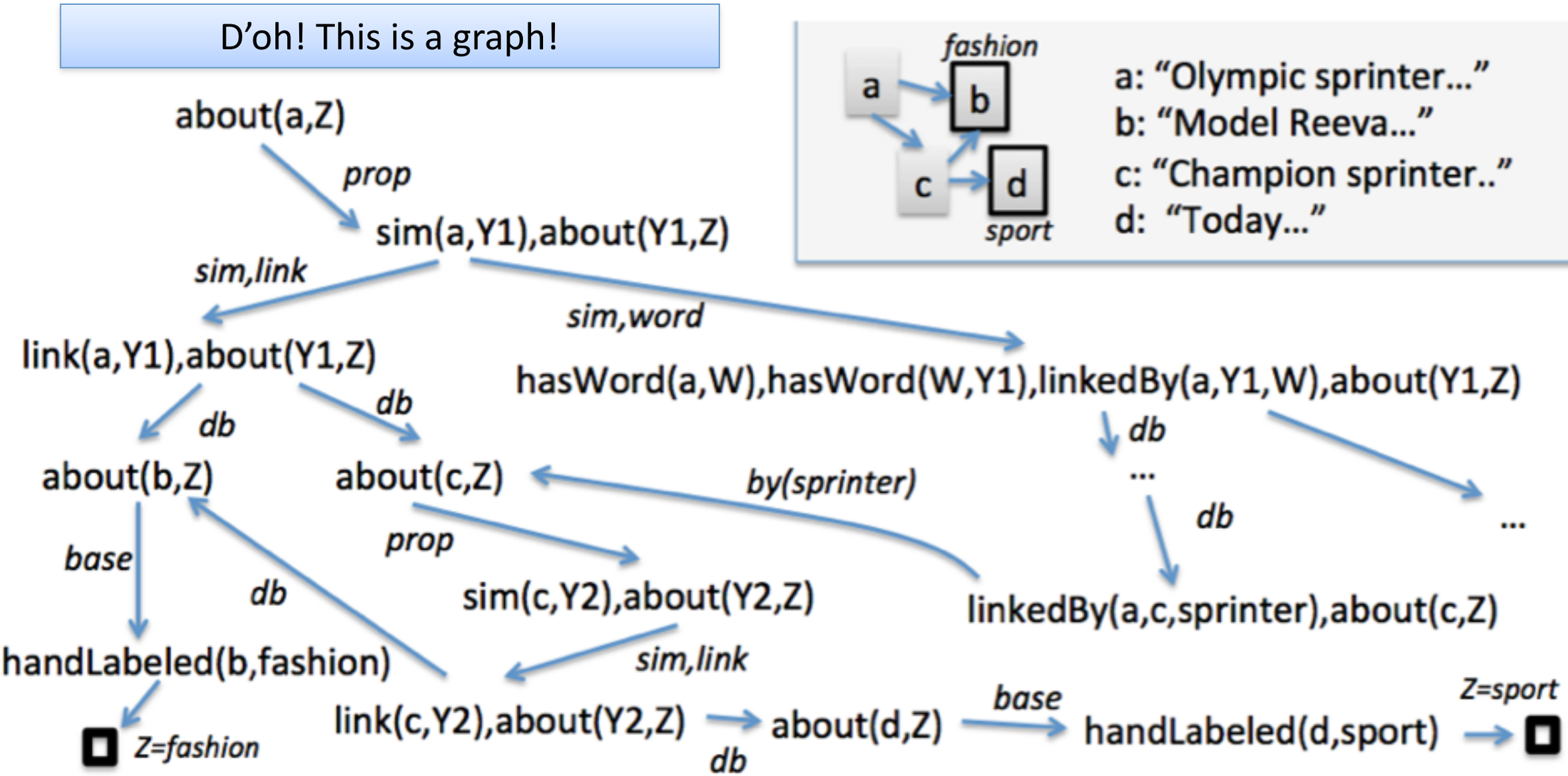


.. and search space…

[Slide by William Cohen]

```
about(X,Z) :- handLabeled(X,Z)          # base.
about(X,Z) :- sim(X,Y),about(Y,Z)       # prop.
sim(X,Y) :- links(X,Y)                  # sim,link.
sim(X,Y) :-
    hasWord(X,W),hasWord(Y,W),
    linkedBy(X,Y,W)                     # sim,word.
linkedBy(X,Y,W) :- true                 # by(W).
```

D'oh! This is a graph!

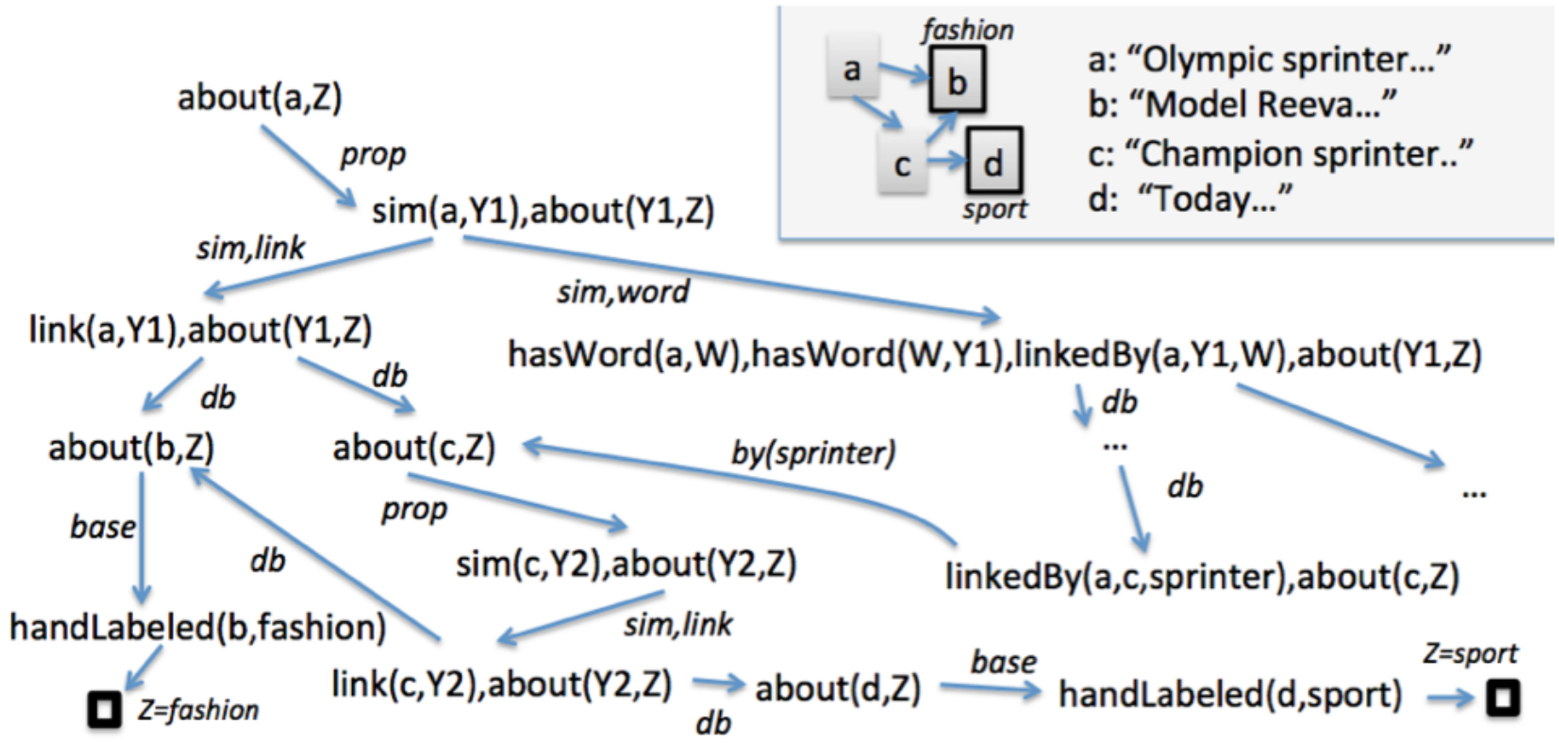


.. and search space...

```
about(X,Z) :- handLabeled(X,Z)          # base.
about(X,Z) :- sim(X,Y),about(Y,Z)       # prop.
sim(X,Y) :- links(X,Y)                  # sim,link.
sim(X,Y) :-
    hasWord(X,W),hasWord(Y,W),
    linkedBy(X,Y,W)                     # sim,word.
linkedBy(X,Y,W) :- true                 # by(W).
```

[Slide by William Cohen]

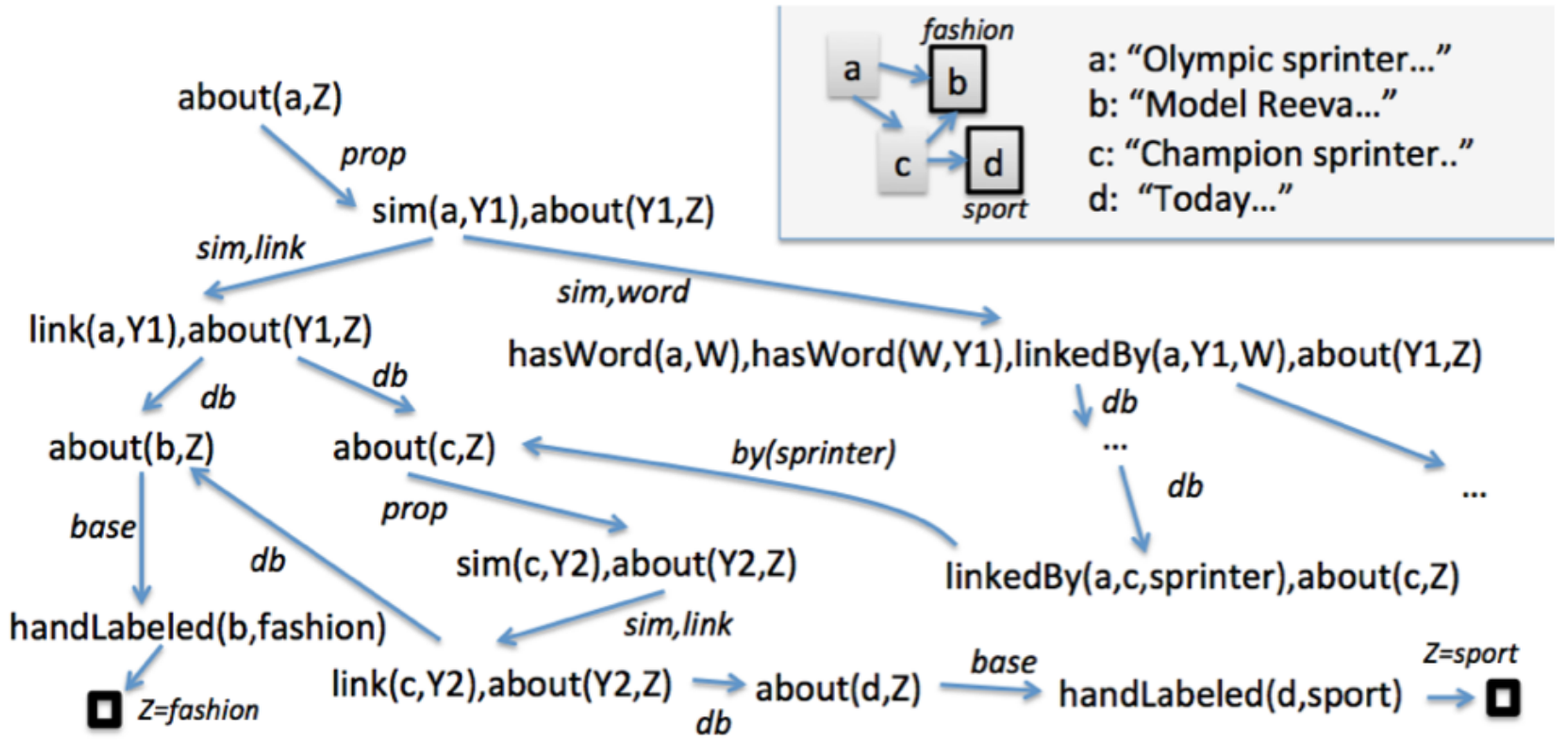
- Score for a query soln (e.g., “Z=sport” for “about(a,Z)”) depends on *probability* of reaching a □ node*
  - *learn* transition probabilities based on features of the rules
  - implicit “reset” transitions with (p≥α) back to query node
- Looking for answers supported by *many short proofs*



[Slide by William Cohen]

- Score for a query soln (e.g., "Z=sport" for "about(a,Z)") depends on *probability* of reaching a □ node*
  - *learn* transition probabilities based on features of the rules
  - implicit "reset" transitions with (p≥α) back to query node
- Looking for answers supported by *many short proofs*

*Exactly as in Stochastic Logic Programs [Cussens, 2001]

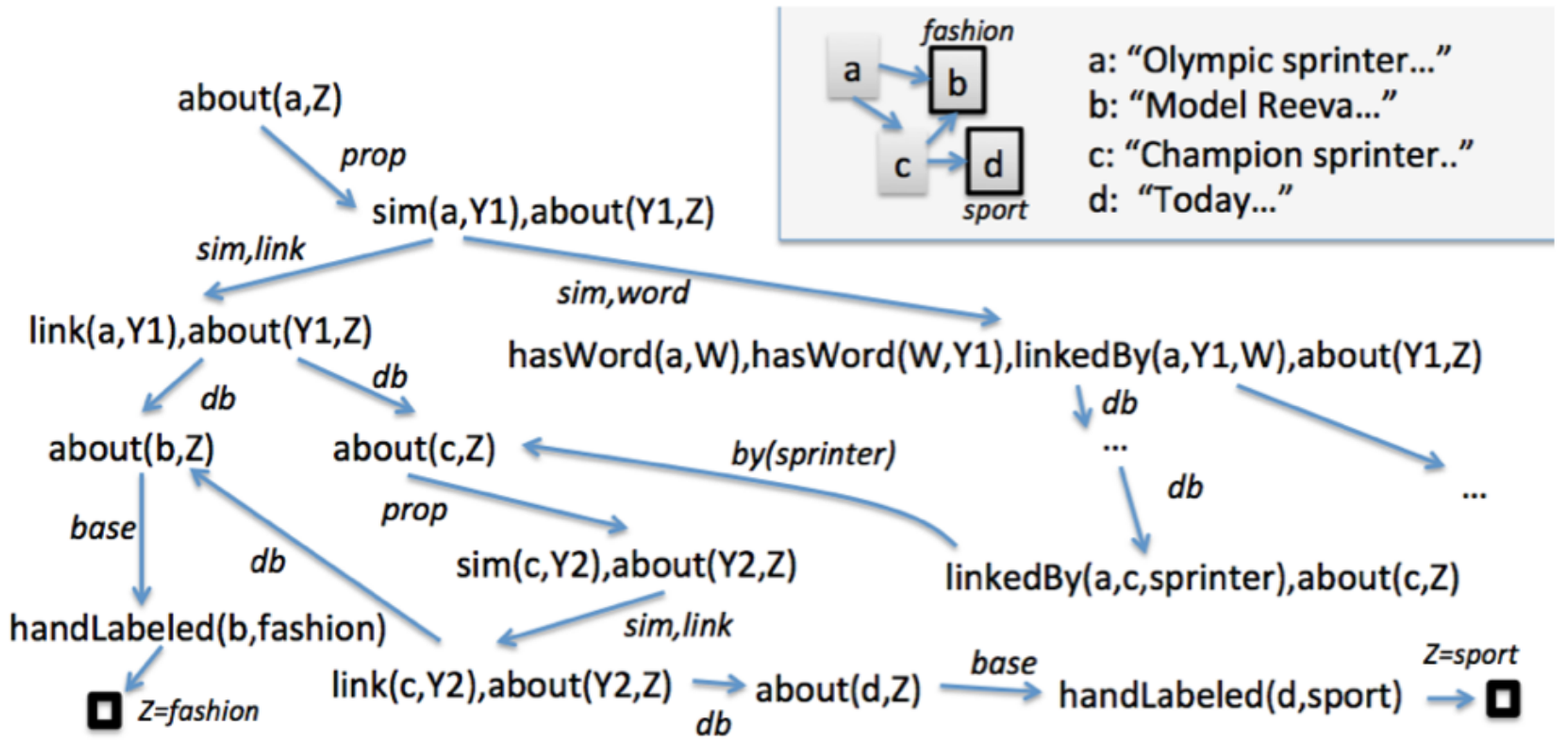


[Slide by William Cohen]

- Score for a query soln (e.g., "Z=sport" for "about(a,Z)") depends on *probability* of reaching a □ node*
  - *learn* transition probabilities based on features of the rules
  - implicit "reset" transitions with (p≥α) back to query node
- Looking for answers supported by *many short proofs*

"Grounding" size is O(1/αε) … ie *independent* of DB size ➔ fast approx incremental inference (Reid,Lang,Chung, 08)

*Exactly as in Stochastic Logic Programs [Cussens, 2001]



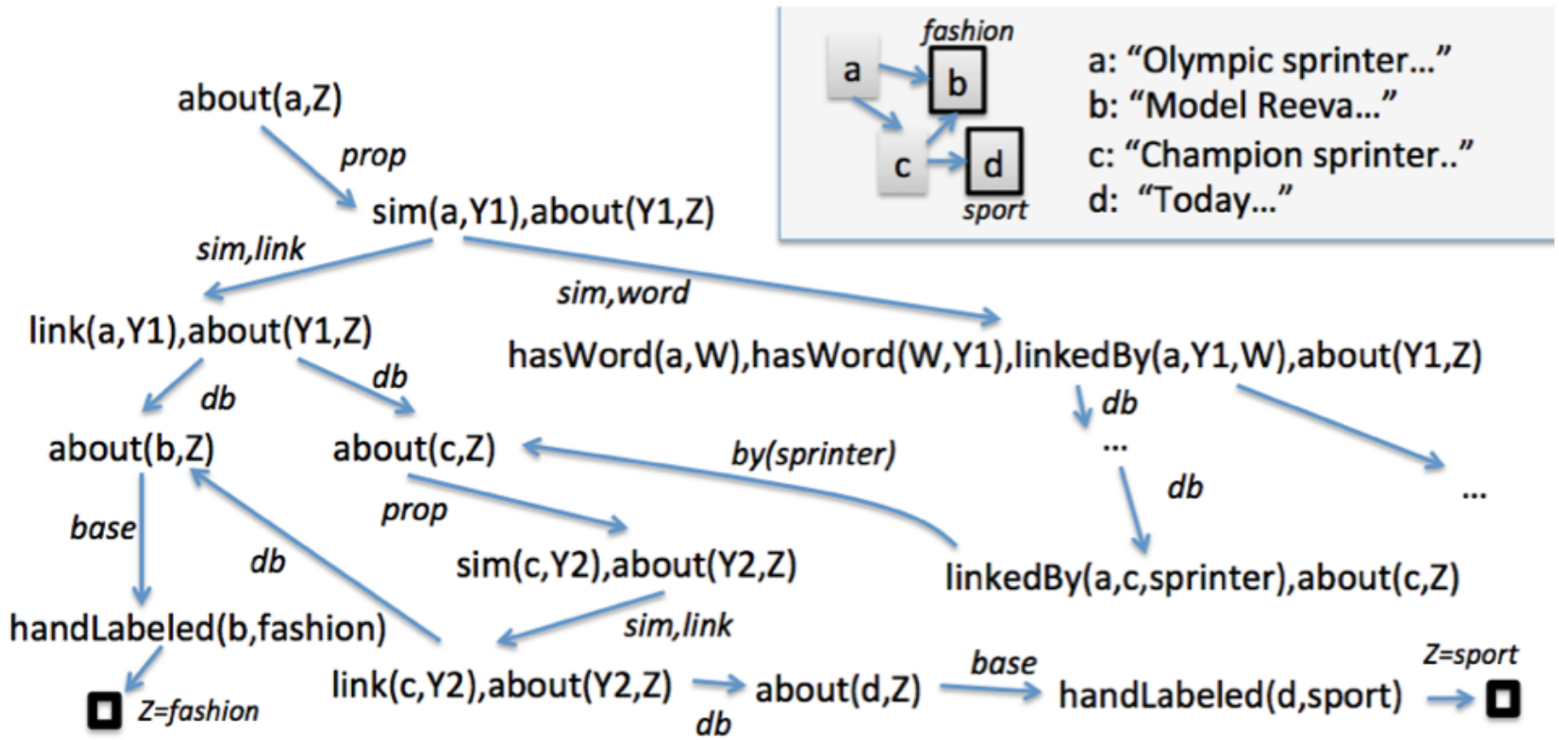
[Slide by William Cohen]

- Score for a query soln (e.g., "Z=sport" for "about(a,Z)") depends on *probability* of reaching a □ node*
  - *learn* transition probabilities based on features of the rules
  - implicit "reset" transitions with (p≥α) back to query node
- Looking for answers supported by *many short proofs*

"Grounding" size is O(1/αε) … ie *independent* of DB size ➔ fast approx incremental inference (Reid,Lang,Chung, 08)

*Exactly as in Stochastic Logic Programs [Cussens, 2001]

Learning: supervised variant of personalized PageRank (Backstrom & Leskovic, 2011)



[Slide by William Cohen]

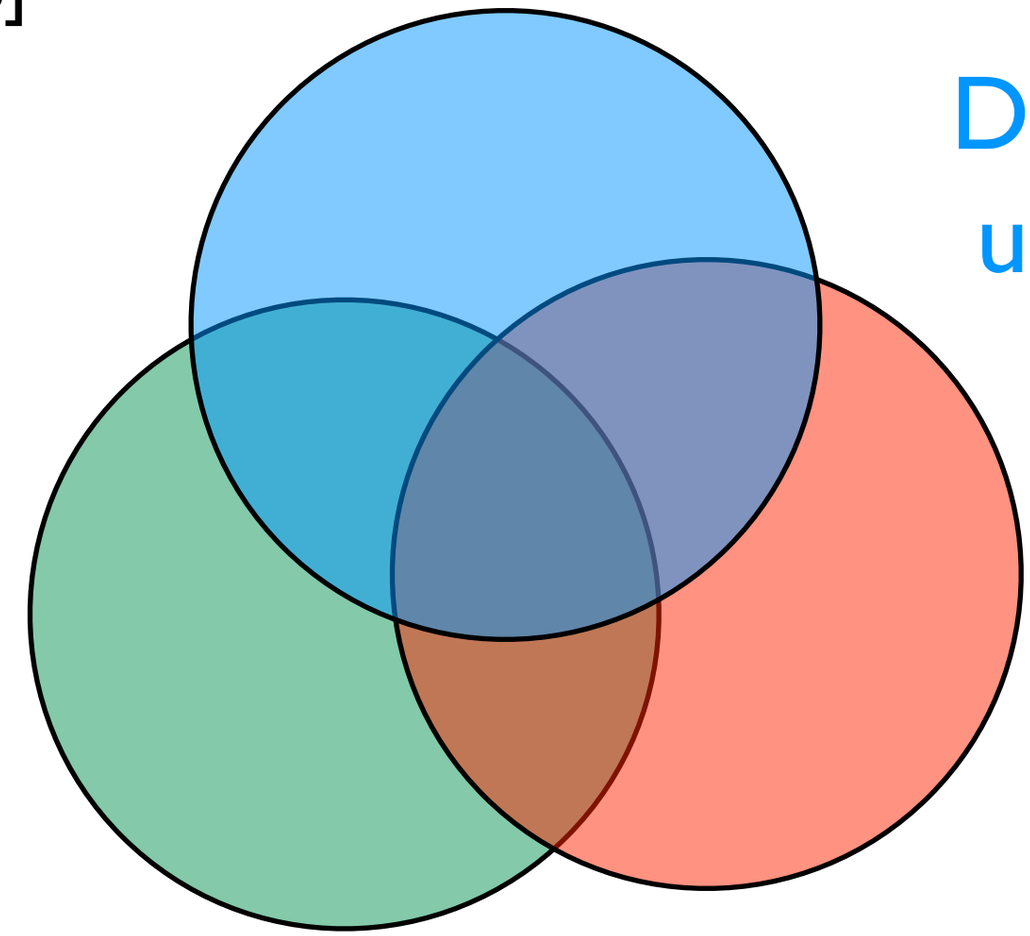
Probabilistic Programming Languages outside LP

- IBAL [Pfeffer 01]
- Figaro [Pfeffer 09]
- Church [Goodman et al 08]
- BLOG [Milch et al 05]
- and many more appearing recently

Church

probabilistic functional programming

[Goodman et al, UAI 08]



Dealing with uncertainty

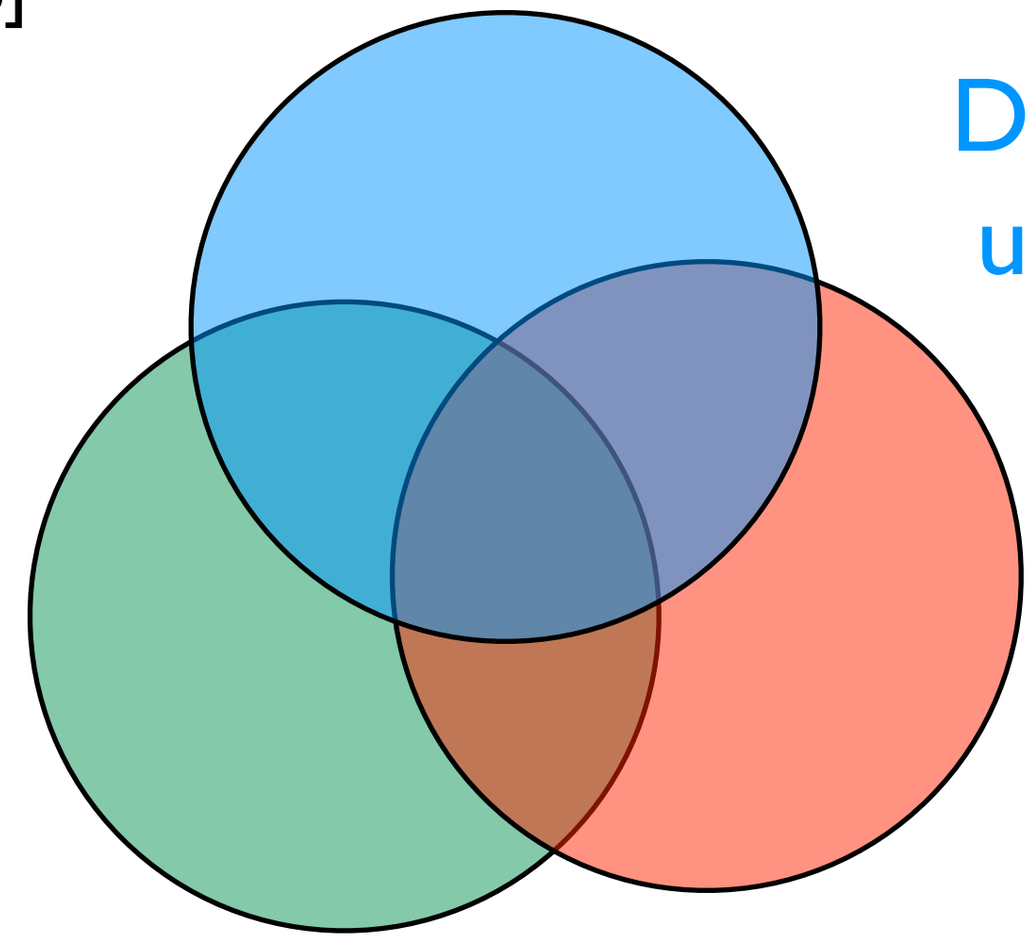
Reasoning with relational data

Learning

Church

probabilistic functional programming

[Goodman et al, UAI 08]



Dealing with uncertainty

functional programming

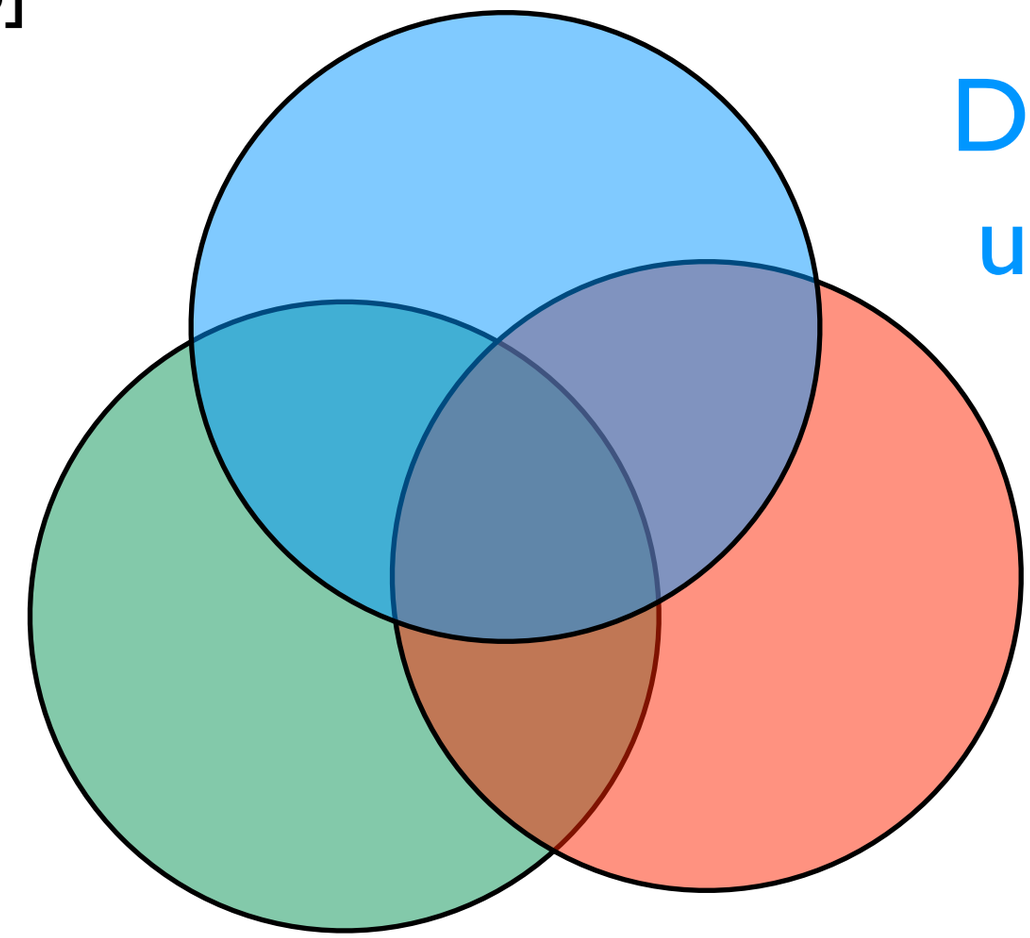
Learning

```
(define plus5 (lambda (x) (+ x 5)))

(map plus5 '(1 2 3))
```

Church

probabilistic functional programming

[Goodman et al, UAI 08]



Dealing with uncertainty

functional programming

Learning

one execution

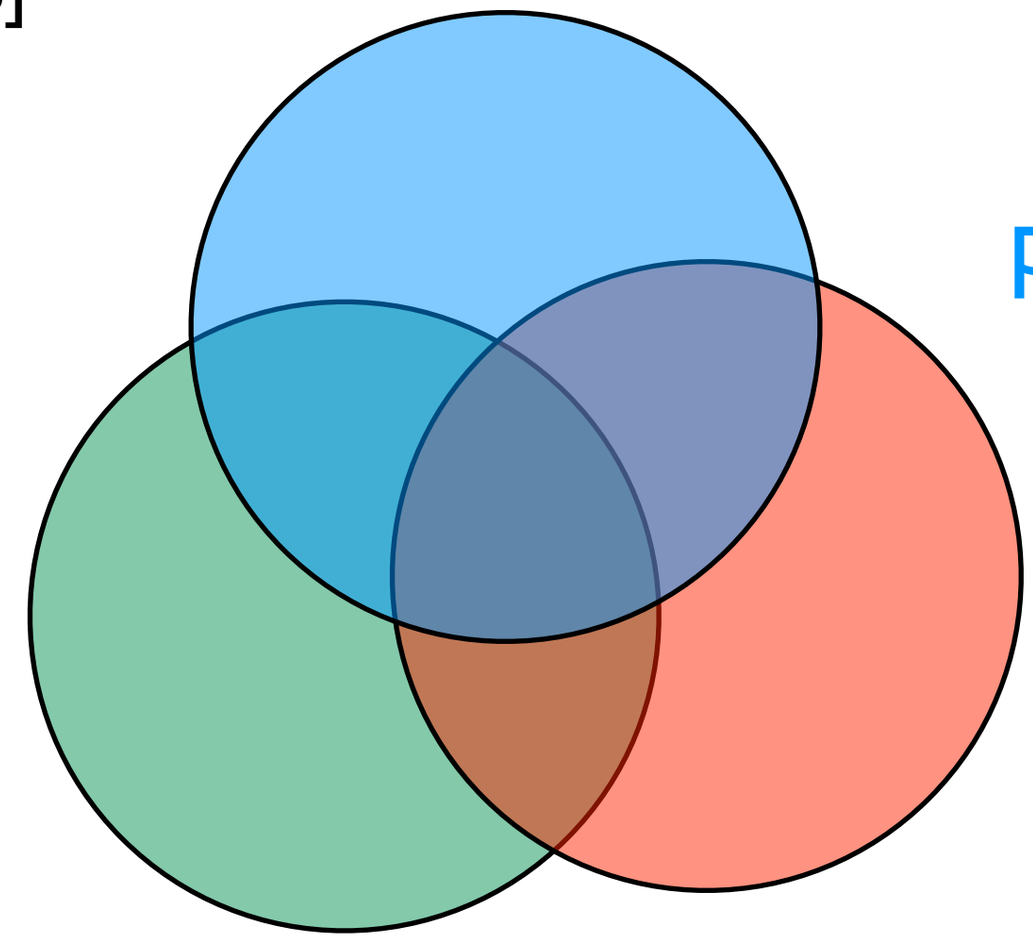
```
(define plus5 (lambda (x) (+ x 5)))

(map plus5 '(1 2 3))
```

Church

probabilistic functional programming

[Goodman et al, UAI 08]

```
(define randplus5
 (lambda (x) (if (flip 0.6)
                 (+ x 5)
                 x)))

(map randplus5 '(1 2 3))
```



random primitives

functional programming

Learning

one execution

```
(define plus5 (lambda (x) (+ x 5)))

(map plus5 '(1 2 3))
```

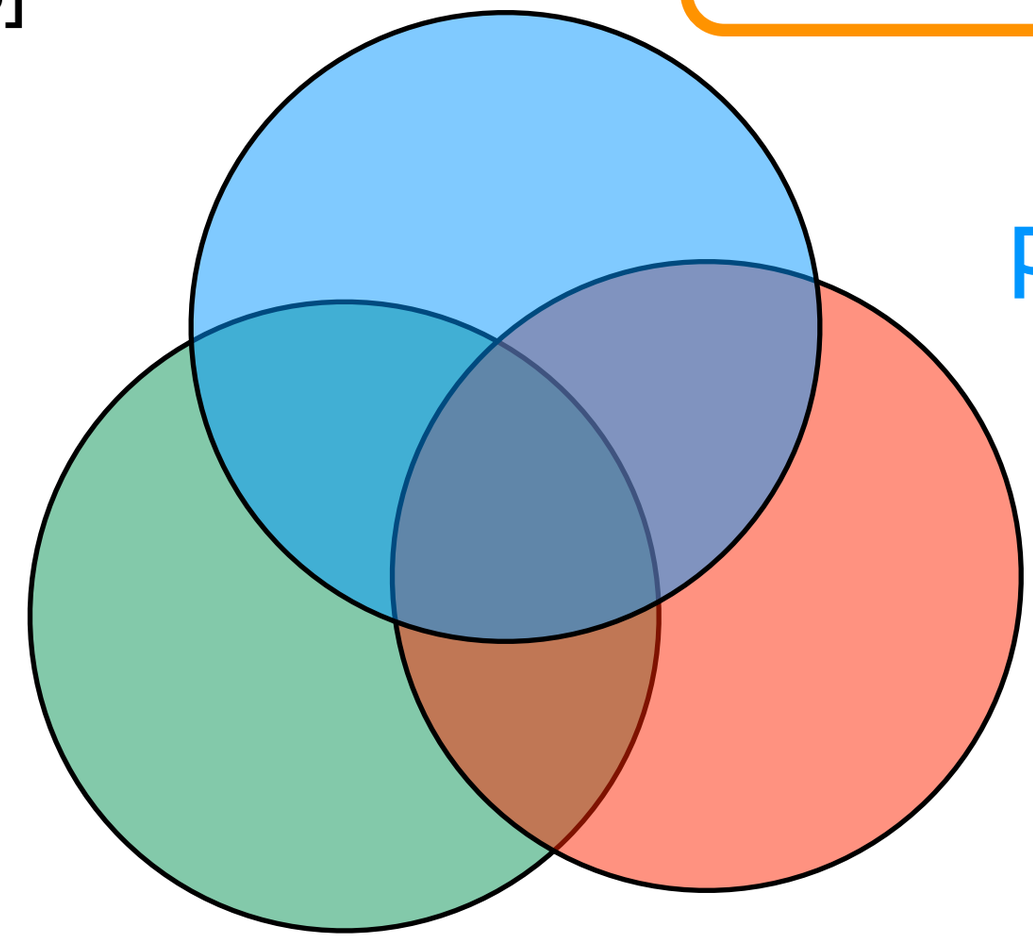
Church

probabilistic functional programming

[Goodman et al, UAI 08]

several possible executions

```
(define randplus5
 (lambda (x) (if (flip 0.6)
                 (+ x 5)
                 x)))

(map randplus5 '(1 2 3))
```



random primitives

functional programming

Learning

one execution

```
(define plus5 (lambda (x) (+ x 5)))

(map plus5 '(1 2 3))
```

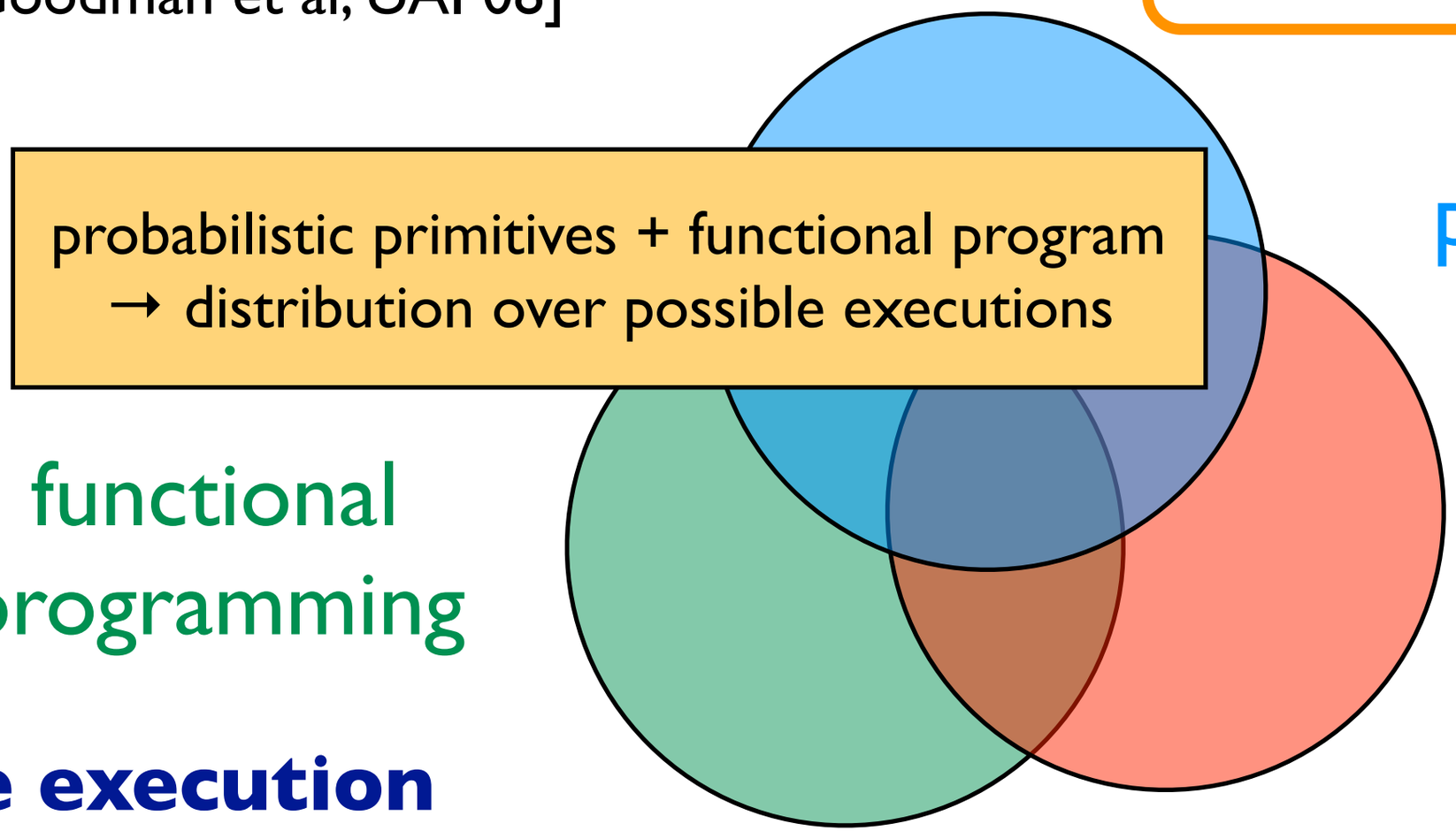
Church

probabilistic functional programming

[Goodman et al, UAI 08]

several possible executions

```
(define randplus5
 (lambda (x) (if (flip 0.6)
                 (+ x 5)
                 x)))

(map randplus5 '(1 2 3))
```



one execution

```
(define plus5 (lambda (x) (+ x 5)))

(map plus5 '(1 2 3))
```

Church vs ProbLog

```
(define randplus5 (lambda (x) (if (flip 0.6) (+ x 5) x)))

(map randplus5 '(1 2))
```

Church vs ProbLog

```
(define randplus5 (lambda (x) (if (flip 0.6) (+ x 5) x)))

(map randplus5 '(1 2))
```

Church result: (1 2) with 0.4×0.4
(1 7) with 0.4×0.6
(6 2) with 0.6×0.4
(6 7) with 0.6×0.6

Church vs ProbLog

```
(define randplus5 (lambda (x) (if (flip 0.6) (+ x 5) x)))

(map randplus5 '(1 2))
```

Church result: (1 2) with 0.4×0.4
(1 7) with 0.4×0.6
(6 2) with 0.6×0.4
(6 7) with 0.6×0.6

```
0.4::p5(N,N);0.6::p5(N,M) :- M is N+5.

lp5([],[]).
lp5([N|L],[M|K]) :-
    p5(N,M),
    lp5(L,K).

query(lp5([1,2],_)).
```

Church vs ProbLog

```
(define randplus5 (lambda (x) (if (flip 0.6) (+ x 5) x)))

(map randplus5 '(1 2))
```

Church result: (1 2) with 0.4×0.4
(1 7) with 0.4×0.6
(6 2) with 0.6×0.4
(6 7) with 0.6×0.6

```
0.4::p5(N,N);0.6::p5(N,M) :- M is N+5.

lp5([],[]).
lp5([N|L],[M|K]) :-
     p5(N,M),
     lp5(L,K).

query(lp5([1,2],_)).
```

ProbLog result: (1 2) with 0.4×0.4
(1 7) with 0.4×0.6
(6 2) with 0.6×0.4
(6 7) with 0.6×0.6

results for [1,1]?

```
(define randplus5 (lambda (x) (if (flip 0.6) (+ x 5) x)))

(map randplus5 '(1 1))
```

```
0.4::p5(N,N);0.6::p5(N,M) :- M is N+5.
lp5([],[]).
lp5([N|L],[M|K]) :-
    p5(N,M),
    lp5(L,K).

query(lp5([1,1],_)).
```

results for [1,1]?

```
(define randplus5 (lambda (x) (if (flip 0.6) (+ x 5) x)))

(map randplus5 '(1 1))
```

Church result: (1 1) with 0.4×0.4
(1 6) with 0.4×0.6
(6 1) with 0.6×0.4
(6 6) with 0.6×0.6

```
0.4::p5(N,N);0.6::p5(N,M) :- M is N+5.
lp5([],[]).
lp5([N|L],[M|K]) :-
    p5(N,M),
    lp5(L,K).

query(lp5([1,1],_)).
```

results for [1,1]?

```
(define randplus5 (lambda (x) (if (flip 0.6) (+ x 5) x)))

(map randplus5 '(1 1))
```

Church result: (1 1) with 0.4×0.4
(1 6) with 0.4×0.6
(6 1) with 0.6×0.4
(6 6) with 0.6×0.6

```
0.4::p5(N,N);0.6::p5(N,M) :- M is N+5.
lp5([],[]).
lp5([N|L],[M|K]) :-
    p5(N,M),
    lp5(L,K).

query(lp5([1,1],_)).
```

ProbLog result: (1 1) with 0.4
(1 6) with 0.0
(6 1) with 0.0
(6 6) with 0.6

results for [1,1]?

```
(define randplus5 (lambda (x) (if (flip 0.6) (+ x 5) x)))

(map randplus5 '(1 1))
```

Church result: (1 1) with 0.4×0.4
(1 6) with 0.4×0.6
(6 1) with 0.6×0.4
(6 6) with 0.6×0.6

```
0.4::p5(N,N);0.6::p5(N,M) :- M is N+5.

lp5([],[]).
lp5([N|L],[M|K]) :-
     p5(N,M),
     lp5(L,K).

query(lp5([1,1],_)).
```

ProbLog result: (1 1) with 0.4
(1 6) with 0.0
(6 1) with 0.0
(6 6) with 0.6

stochastic memoization

Solution

```
(define randplus5 (lambda (x) (if (flip 0.6) (+ x 5) x)))

(map randplus5 '(1 1))
```

```
0.4::p5(N,N,ID);0.6::p5(N,M,ID) :- M is N+5.
lp5([],[]).
lp5([N|L],[M|K]) :-
    p5(N,M,L),
    lp5(L,K).

query(lp5([1,1],_)).
```

Solution

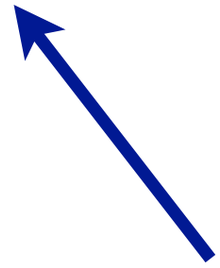
```
(define randplus5 (lambda (x) (if (flip 0.6) (+ x 5) x)))

(map randplus5 '(1 1))
```

```
0.4::p5(N,N,ID);0.6::p5(N,M,ID) :- M is N+5.
lp5([],[]).
lp5([N|L],[M|K]) :-
     p5(N,M,L),
     lp5(L,K).

query(lp5([1,1],_)).
```

identifier distinguishes calls

Stochastic Memoization

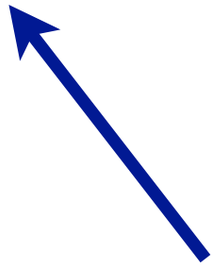
```
(define randplus5 (mem (lambda (x) (if (flip 0.6) (+ x 5) x))))

(map randplus5 '(1 1))
```



remember first value &
reuse for all later calls

Stochastic Memoization

```
(define randplus5 (mem (lambda (x) (if (flip 0.6) (+ x 5) x))))

(map randplus5 '(1 1))
```

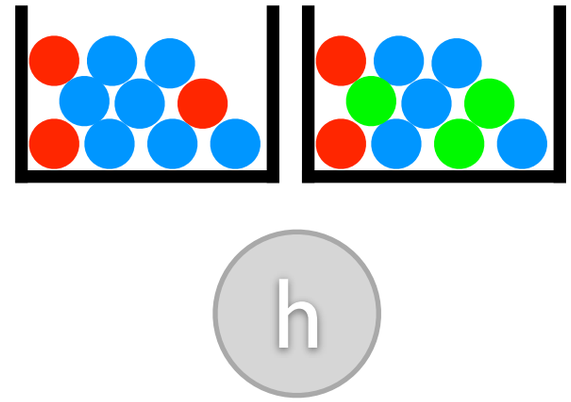


remember first value &
reuse for all later calls

ProbLog always memoizes
PRISM never memoizes
Church allows fine-grained choice

Church by example:

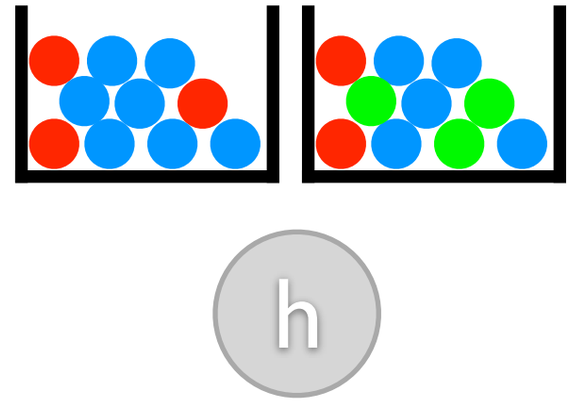
A bit of gambling



- toss (biased) coin & draw ball from each urn
- win if (heads and a red ball) or (two balls of same color)

Church by example:

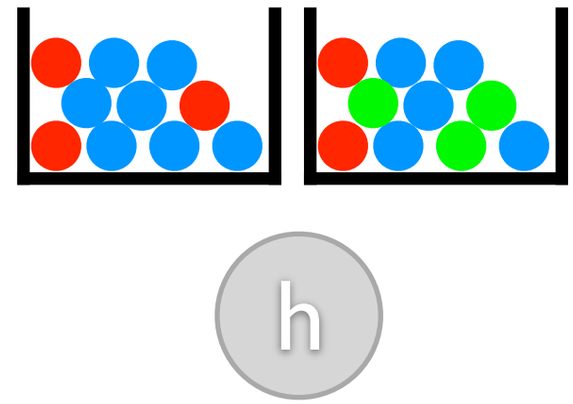
A bit of gambling



- toss (biased) coin & draw ball from each urn
- win if (heads and a red ball) or (two balls of same color)

Church by example:

A bit of gambling

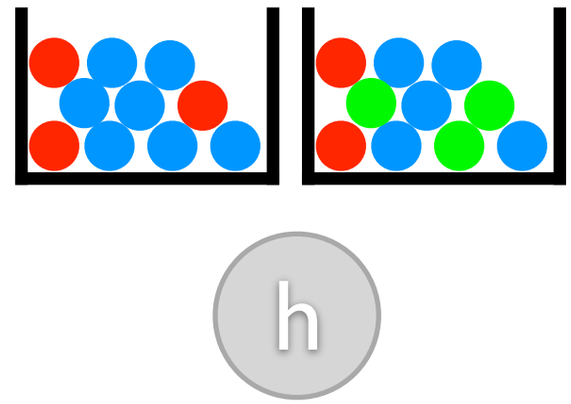


- toss (biased) coin & draw ball from each urn
- win if (heads and a red ball) or (two balls of same color)

```
(define heads (mem (lambda () (flip 0.4))))
```

Church by example:

A bit of gambling

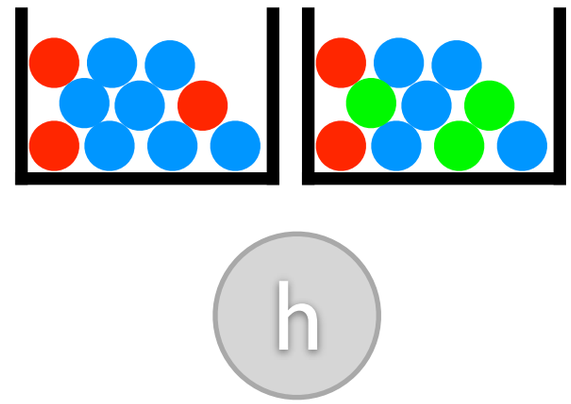


- toss (biased) coin & draw ball from each urn
- win if (heads and a red ball) or (two balls of same color)

```
(define heads (mem (lambda () (flip 0.4))))
```

Church by example:

A bit of gambling

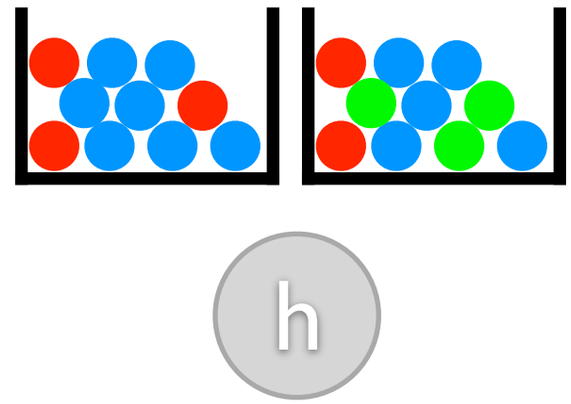


- toss (biased) coin & draw ball from each urn
- win if (heads and a red ball) or (two balls of same color)

```
(define heads (mem (lambda () (flip 0.4))))

(define color1 (mem (lambda () (if (flip 0.3) 'red 'blue))))
```

Church by example:

A bit of gambling

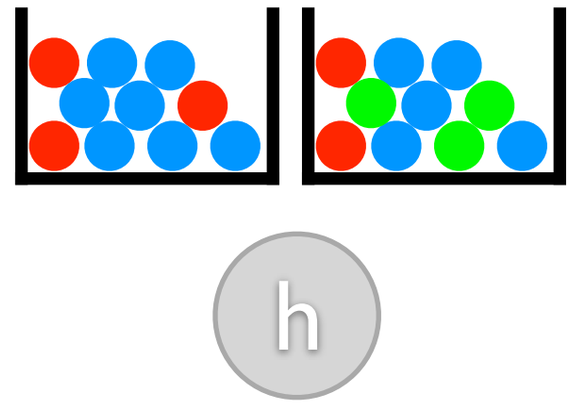


- toss (biased) coin & draw ball from each urn
- win if (heads and a red ball) or (two balls of same color)

```
(define heads (mem (lambda () (flip 0.4))))

(define color1 (mem (lambda () (if (flip 0.3) 'red 'blue))))

(define color2 (mem (lambda ()
                      (multinomial '(red green blue) '(0.2 0.3 0.5)))))
```

Church by example:

A bit of gambling

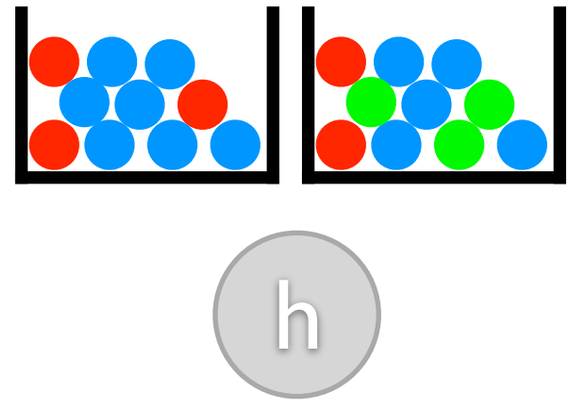


- toss (biased) coin & draw ball from each urn
- win if (heads and a red ball) or (two balls of same color)

```
(define heads (mem (lambda () (flip 0.4))))

(define color1 (mem (lambda () (if (flip 0.3) 'red 'blue))))

(define color2 (mem (lambda ()
                  (multinomial '(red green blue) '(0.2 0.3 0.5)))))
```

Church by example:

A bit of gambling

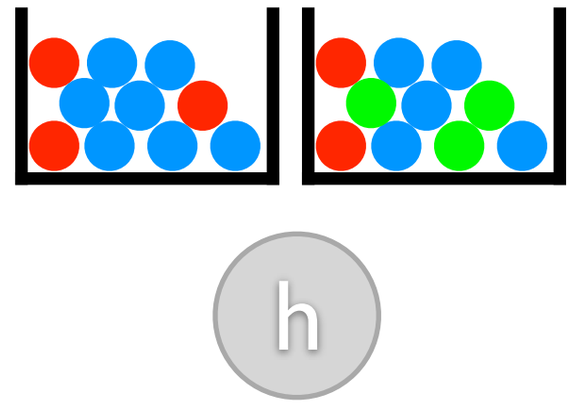


- toss (biased) coin & draw ball from each urn
- win if (heads and a red ball) or (two balls of same color)

```
(define heads (mem (lambda () (flip 0.4))))

(define color1 (mem (lambda () (if (flip 0.3) 'red 'blue))))

(define color2 (mem (lambda ()
                      (multinomial '(red green blue) '(0.2 0.3 0.5)))))

(define redball (or (equal? (color1) 'red) (equal? (color2) 'red)))
```

Church by example:

A bit of gambling

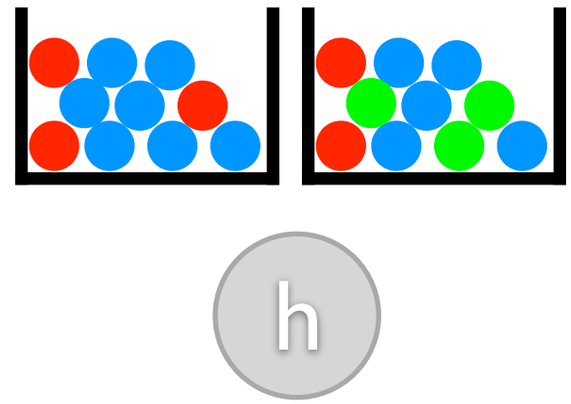


- toss (biased) coin & draw ball from each urn
- win if (heads and a red ball) or (two balls of same color)

```
(define heads (mem (lambda () (flip 0.4))))

(define color1 (mem (lambda () (if (flip 0.3) 'red 'blue))))

(define color2 (mem (lambda ()
                      (multinomial '(red green blue) '(0.2 0.3 0.5)))))

(define redball (or (equal? (color1) 'red) (equal? (color2) 'red)))

(define win1 (and (heads) redball))
```

Church by example:

A bit of gambling

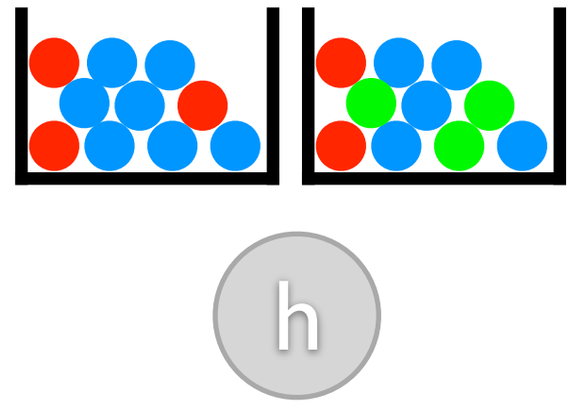


- toss (biased) coin & draw ball from each urn
- win if (heads and a red ball) or (two balls of same color)

```
(define heads (mem (lambda () (flip 0.4))))

(define color1 (mem (lambda () (if (flip 0.3) 'red 'blue))))

(define color2 (mem (lambda ()
                      (multinomial '(red green blue) '(0.2 0.3 0.5)))))

(define redball (or (equal? (color1) 'red) (equal? (color2) 'red)))

(define win1 (and (heads) redball))
```

Church by example:

A bit of gambling

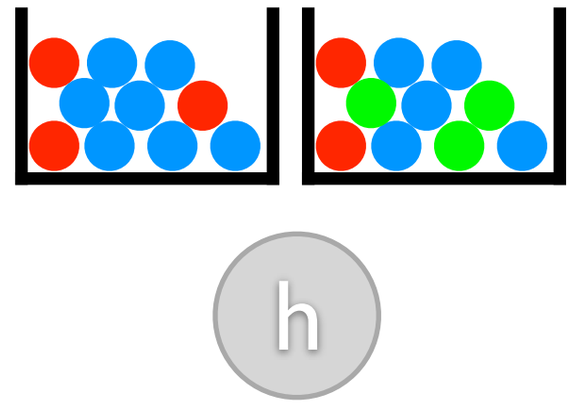


- toss (biased) coin & draw ball from each urn
- win if (heads and a red ball) or (two balls of same color)

```
(define heads (mem (lambda () (flip 0.4))))

(define color1 (mem (lambda () (if (flip 0.3) 'red 'blue))))

(define color2 (mem (lambda ()
                      (multinomial '(red green blue) '(0.2 0.3 0.5)))))

(define redball (or (equal? (color1) 'red) (equal? (color2) 'red)))

(define win1 (and (heads) redball))

(define win2 (equal? (color1) (color2)))
```

Church by example:

A bit of gambling

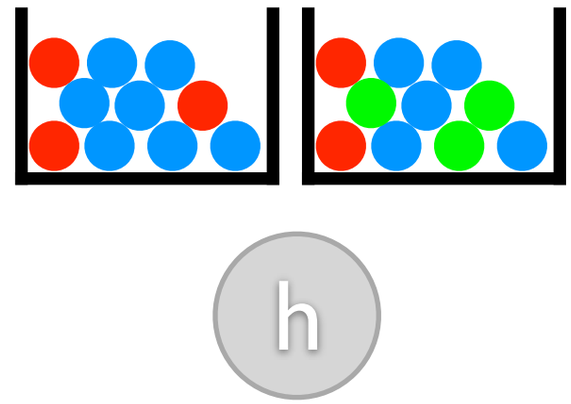


- toss (biased) coin & draw ball from each urn
- win if (heads and a red ball) or (two balls of same color)

```
(define heads (mem (lambda () (flip 0.4))))

(define color1 (mem (lambda () (if (flip 0.3) 'red 'blue))))

(define color2 (mem (lambda ()
                      (multinomial '(red green blue) '(0.2 0.3 0.5)))))

(define redball (or (equal? (color1) 'red) (equal? (color2) 'red)))

(define win1 (and (heads) redball))

(define win2 (equal? (color1) (color2)))
```

Church by example:

A bit of gambling

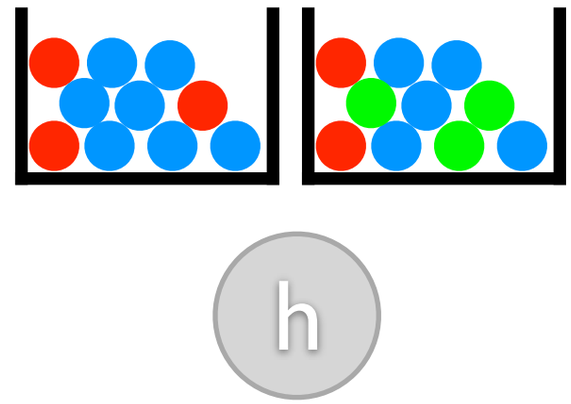


- toss (biased) coin & draw ball from each urn
- win if (heads and a red ball) or (two balls of same color)

```
(define heads (mem (lambda () (flip 0.4))))

(define color1 (mem (lambda () (if (flip 0.3) 'red 'blue))))

(define color2 (mem (lambda ()
                      (multinomial '(red green blue) '(0.2 0.3 0.5)))))

(define redball (or (equal? (color1) 'red) (equal? (color2) 'red)))

(define win1 (and (heads) redball))

(define win2 (equal? (color1) (color2)))

(define win (or win1 win2))
```

Church by example:

A bit of gambling



- toss (biased) coin & draw ball from each urn
- win if (heads and a red ball) or (two balls of same color)

```
(define heads (mem (lambda () (flip 0.4))))

(define color1 (mem (lambda () (if (flip 0.3) 'red 'blue))))

(define color2 (mem (lambda ()
                      (multinomial '(red green blue) '(0.2 0.3 0.5)))))

(define redball (or (equal? (color1) 'red) (equal? (color2) 'red)))

(define win1 (and (heads) redball))

(define win2 (equal? (color1) (color2)))

(define win (or win1 win2))
```

Sampling execution

```
(define heads (mem (lambda () (flip 0.4))))

(define color1 (mem (lambda () (if (flip 0.3) 'red 'blue))))

(define color2 (mem (lambda ()
                      (multinomial '(red green blue) '(0.2 0.3 0.5)))))

(define redball (or (equal? (color1) 'red) (equal? (color2) 'red)))

(define win1 (and (heads) redball))

(define win2 (equal? (color1) (color2)))

(define win (or win1 win2))
```

```
win
```

query

Marginals via enumeration

```
(enumeration-query
  (define heads (mem (lambda () (flip 0.4))))
  (define color1 (mem (lambda () (if (flip 0.3) 'red 'blue))))
  (define color2 (mem (lambda ()
                        (multinomial '(red green blue) '(0.2 0.3 0.5)))))
(define redball (or (equal? (color1) 'red) (equal? (color2) 'red)))
(define win1 (and (heads) redball))
(define win2 (equal? (color1) (color2)))
(define win (or win1 win2))

win ← query

true )
  ← evidence
```

Histogram via sampling

```
(hist

(repeat 1000 (lambda ()
               (rejection-query
(define heads (mem (lambda () (flip 0.4))))

(define color1 (mem (lambda () (if (flip 0.3) 'red 'blue))))

(define color2 (mem (lambda ()
                      (multinomial '(red green blue) '(0.2 0.3 0.5)))))

(define redball (or (equal? (color1) 'red) (equal? (color2) 'red)))

(define win1 (and (heads) redball))

(define win2 (equal? (color1) (color2)))

(define win (or win1 win2))

win
true ))))
```

query

evidence

Probabilistic Programming Summary

- Church: functional programming + random primitives
  - probabilistic generative model
  - stochastic memoization
  - sampling
  - increasing number of probabilistic programming languages using various underlying paradigms

ProbLog	PRISM	Church
probabilistic facts & choices	probabilistic choices	random primitives
all RVs memoized	no RVs memoized	user-defined per RV
Prolog	Prolog with mutually exclusive derivations	λ-calculus functions
distribution over worlds	distribution over derivations / answers	distribution over computations / answers

Roadmap

- Modeling
- Inference
- Learning & Dynamics
- Advanced Inference and KBMC

... with some detours on the way

PART II : Inference

Inference / Reasoning

- Most of the work in PP and StarAI is on inference
- It is hard (complexity wise)
- Many inference methods
  - exact, approximate, sampling and lifted …

Answering Questions

Given:

program

queries

evidence

?

Find:

marginal probabilities

conditional probabilities

MPE state

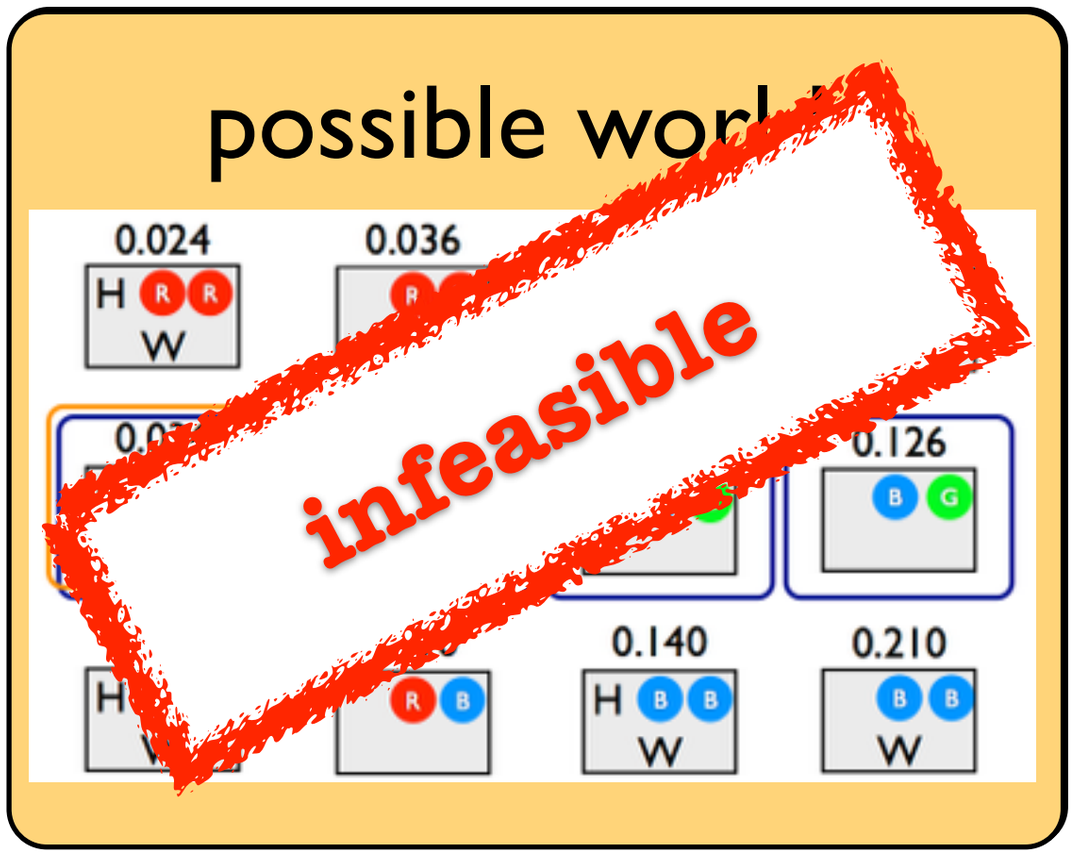
Answering Questions

Given:

program

queries

evidence



Find:

marginal probabilities

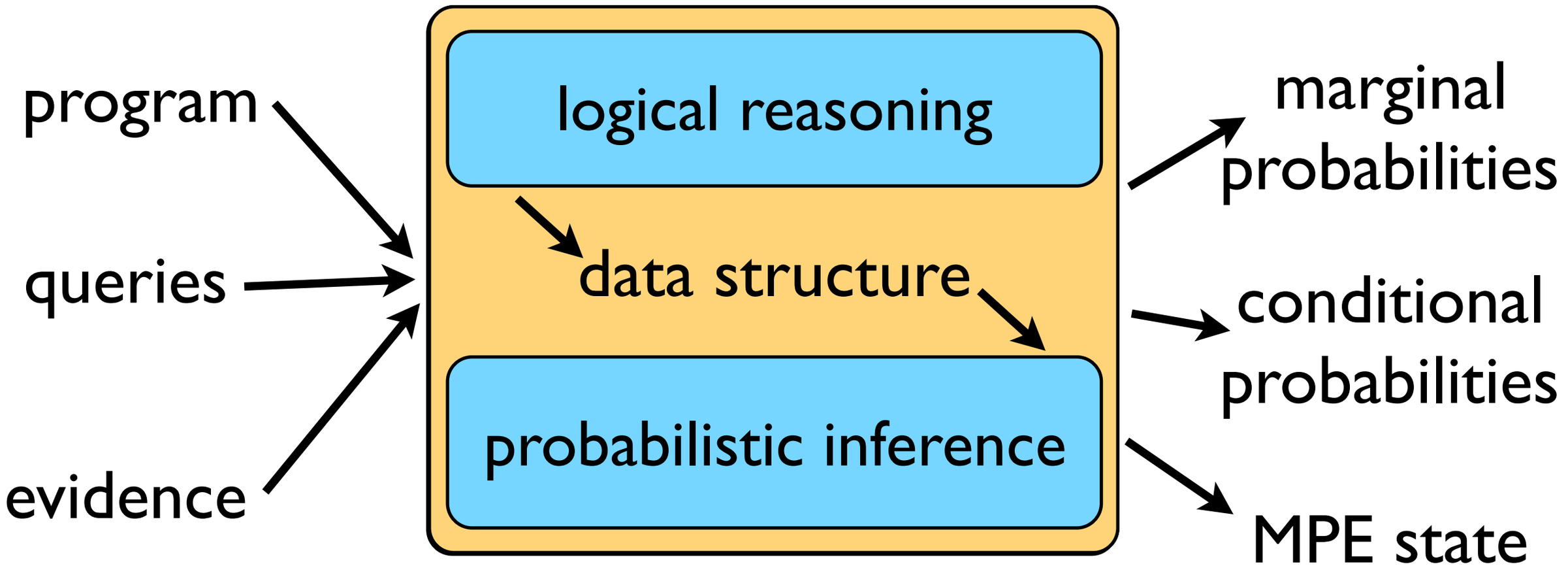
conditional probabilities

MPE state

Answering Questions

Given:

Find:



Two Steps

- **Logical inference -**
  - about a ground logical theory
    - proofs or model theoretic …
      - *Result: Weighted Model Counting problem*
- **Probabilistic propositional inference** —
  - Backtracking search — DPLL, VE, RC based
  - Knowledge Compilation
- **Advanced** — lifted inference

Logical Reasoning: Proofs in Prolog

```
stress(ann).
influences(ann,bob).
influences(bob,carl).

smokes(X) :- stress(X).
smokes(X) :-
      influences(Y,X),
      smokes(Y).
```

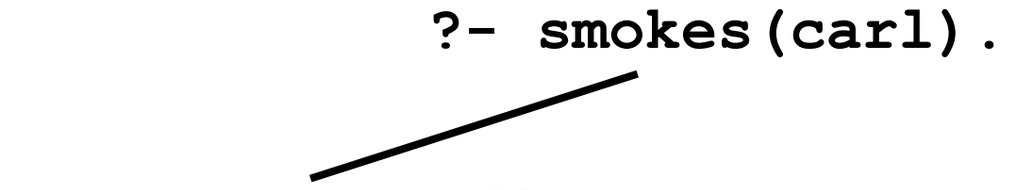
Logical Reasoning: Proofs in Prolog

```
?- smokes(carl).
```

```
stress(ann).
influences(ann,bob).
influences(bob,carl).

smokes(X) :- stress(X).
smokes(X) :-
      influences(Y,X),
      smokes(Y).
```

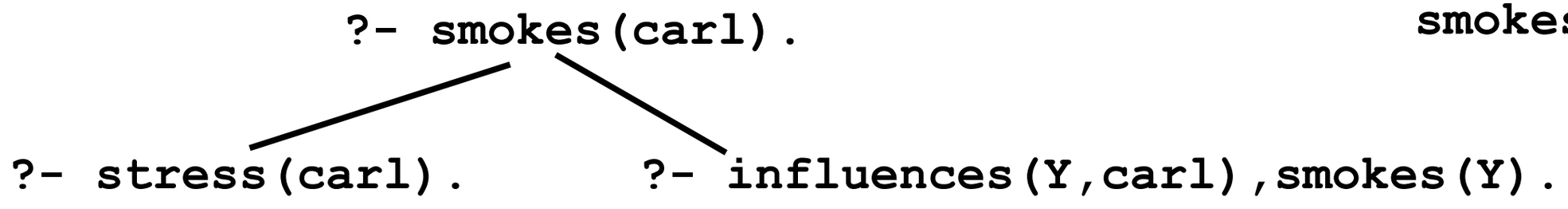
Logical Reasoning: Proofs in Prolog

```
stress(ann).
influences(ann,bob).
influences(bob,carl).

smokes(X) :- stress(X).
smokes(X) :-
      influences(Y,X),
      smokes(Y).
```



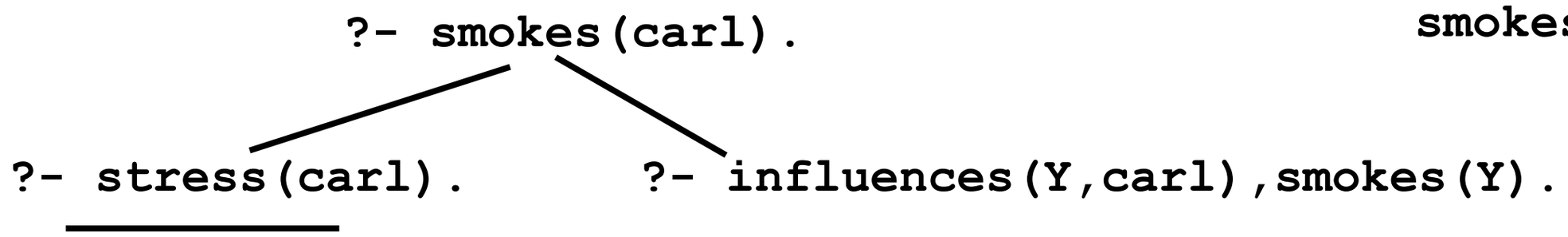
Logical Reasoning: Proofs in Prolog

```
stress(ann).
influences(ann,bob).
influences(bob,carl).

smokes(X) :- stress(X).
smokes(X) :-
      influences(Y,X),
      smokes(Y).
```



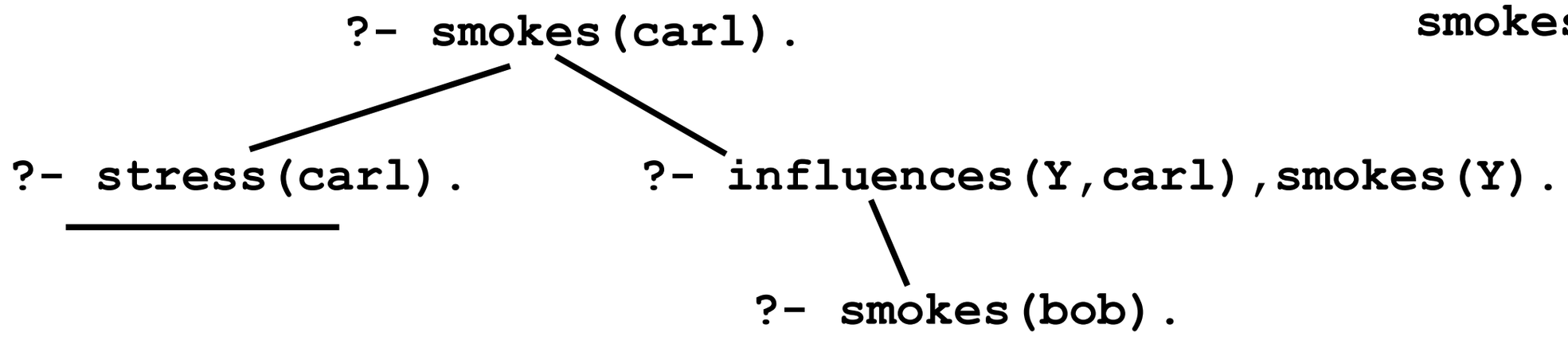
Logical Reasoning: Proofs in Prolog

```
stress(ann).
influences(ann,bob).
influences(bob,carl).

smokes(X)  :- stress(X).
smokes(X)  :-
      influences(Y,X),
      smokes(Y).
```



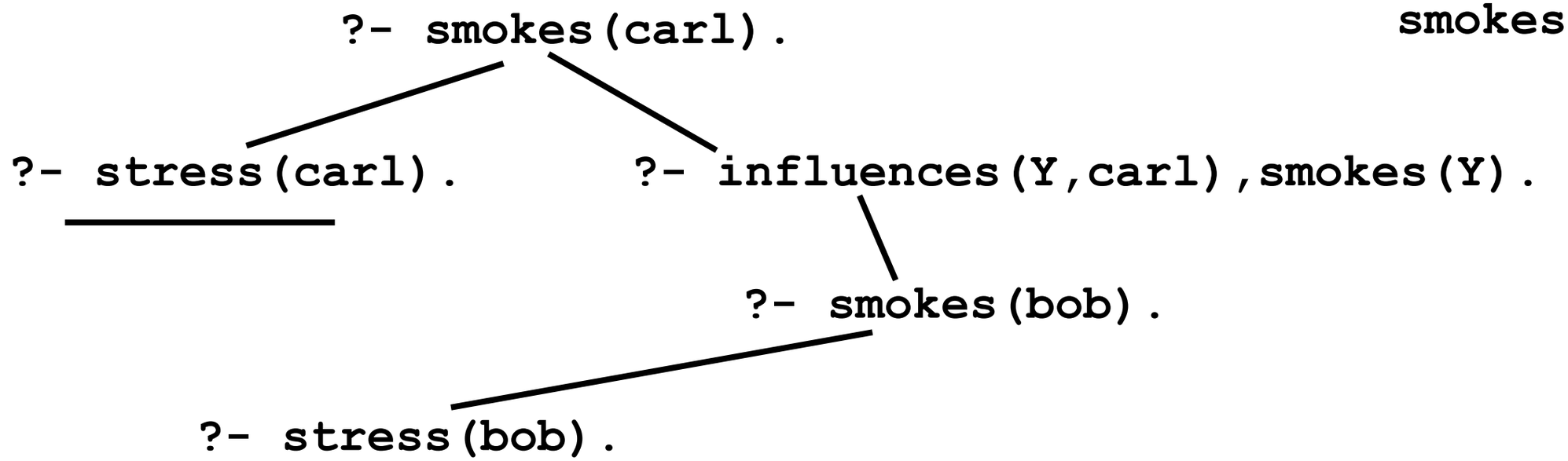
Logical Reasoning: Proofs in Prolog

```
stress(ann).
influences(ann,bob).
influences(bob,carl).

smokes(X)  :- stress(X).
smokes(X)  :-
      influences(Y,X),
      smokes(Y).
```



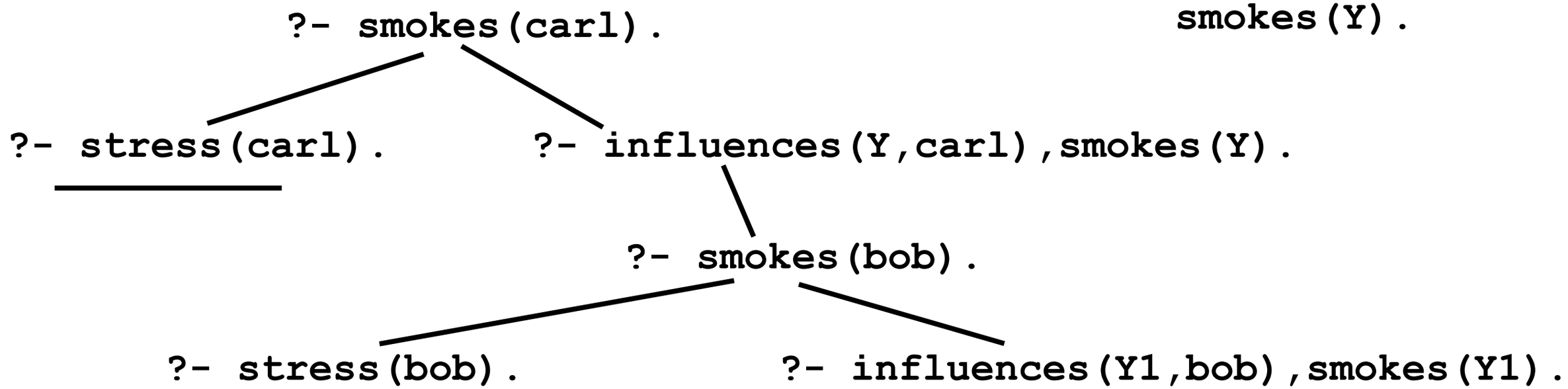
Logical Reasoning: Proofs in Prolog

```
stress(ann).
influences(ann,bob).
influences(bob,carl).

smokes(X) :- stress(X).
smokes(X) :-
      influences(Y,X),
      smokes(Y).
```



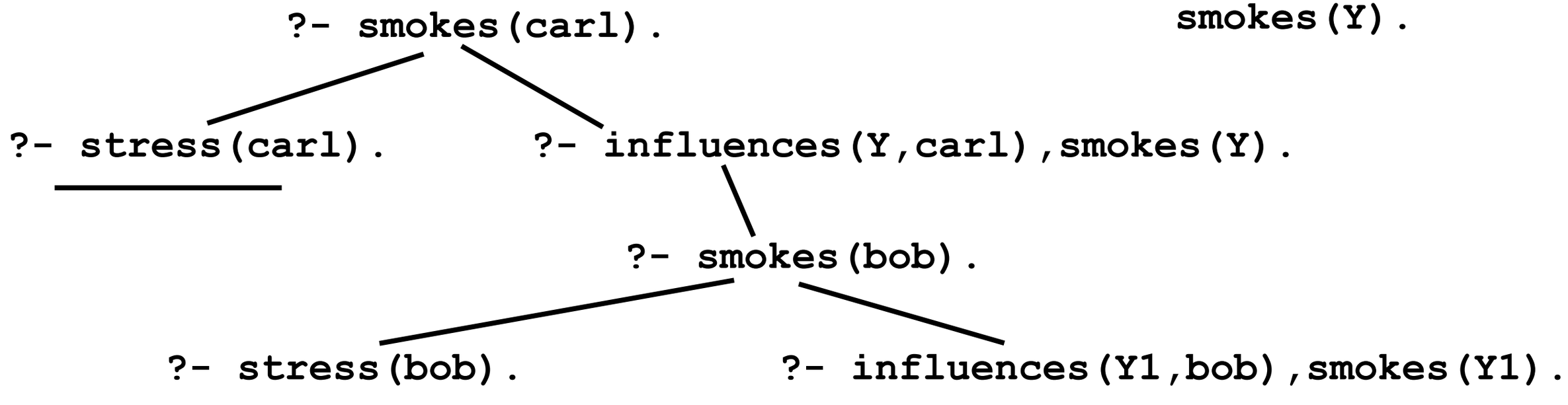
Logical Reasoning: Proofs in Prolog

```
stress(ann).
influences(ann,bob).
influences(bob,carl).

smokes(X)  :- stress(X).
smokes(X)  :-
      influences(Y,X),
      smokes(Y).
```



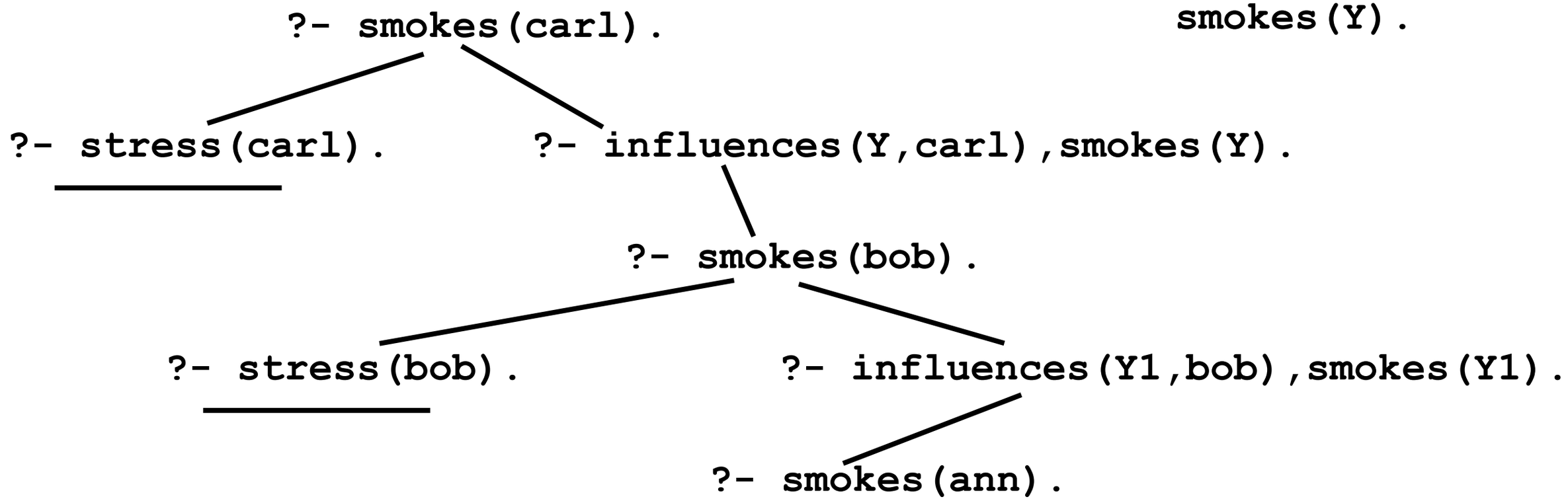
Logical Reasoning: Proofs in Prolog

```
stress(ann).
influences(ann,bob).
influences(bob,carl).

smokes(X) :- stress(X).
smokes(X) :-
      influences(Y,X),
      smokes(Y).
```



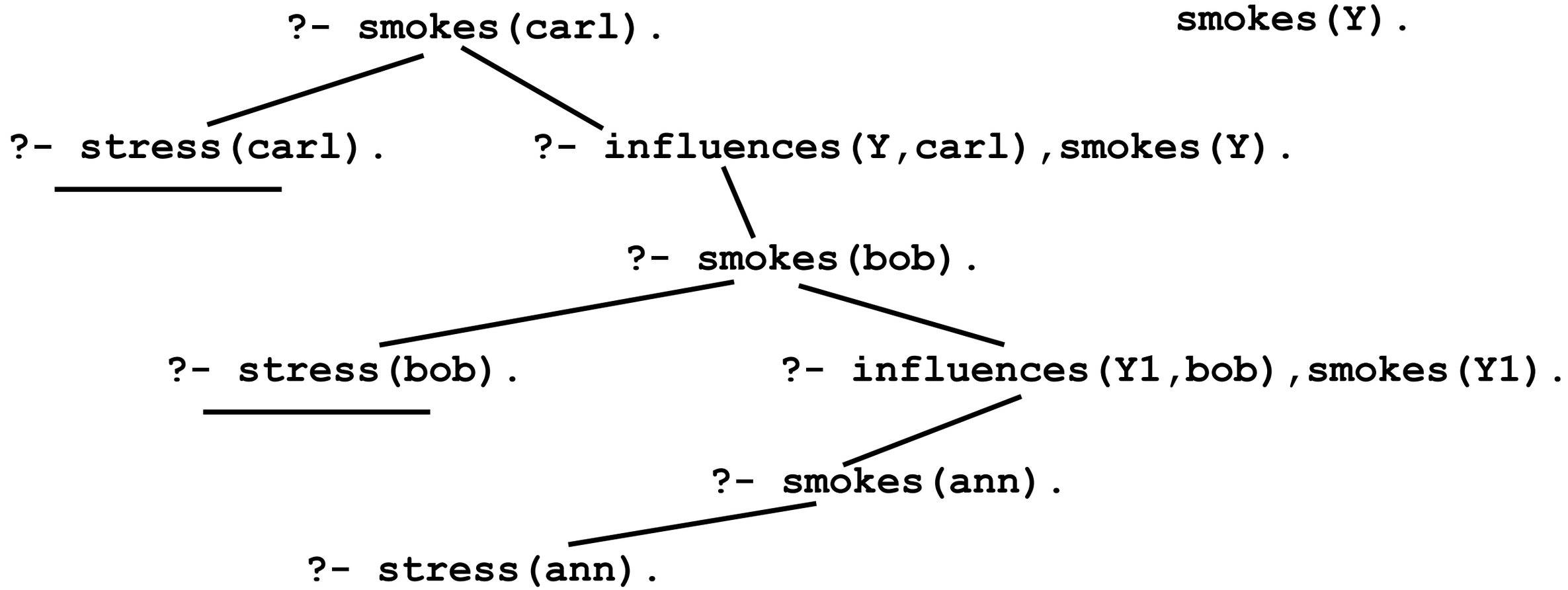
Logical Reasoning: Proofs in Prolog

```
stress(ann).
influences(ann,bob).
influences(bob,carl).

smokes(X)  :- stress(X).
smokes(X)  :-
      influences(Y,X),
      smokes(Y).
```



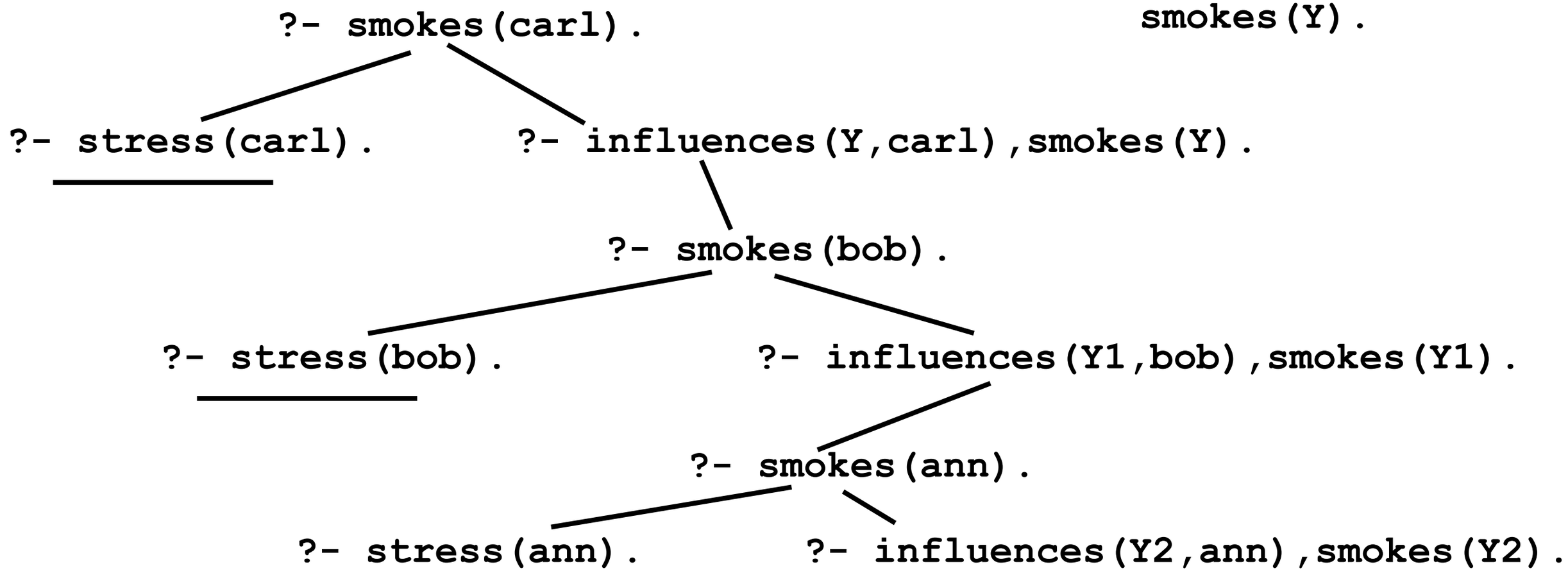
Logical Reasoning: Proofs in Prolog

```
stress(ann).
influences(ann,bob).
influences(bob,carl).

smokes(X)  :- stress(X).
smokes(X)  :-
      influences(Y,X),
      smokes(Y).
```



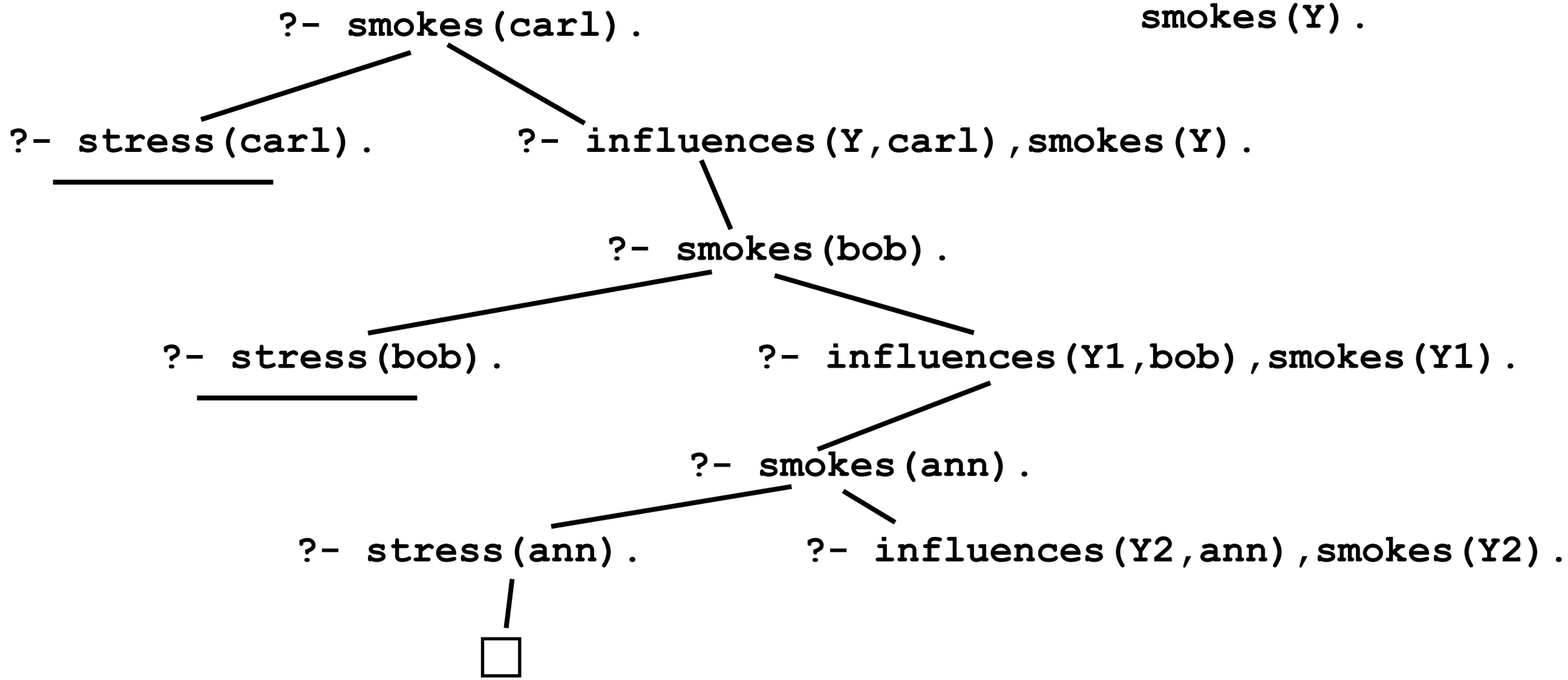
Logical Reasoning: Proofs in Prolog

```
stress(ann).
influences(ann,bob).
influences(bob,carl).

smokes(X)  :- stress(X).
smokes(X)  :-
      influences(Y,X),
      smokes(Y).
```



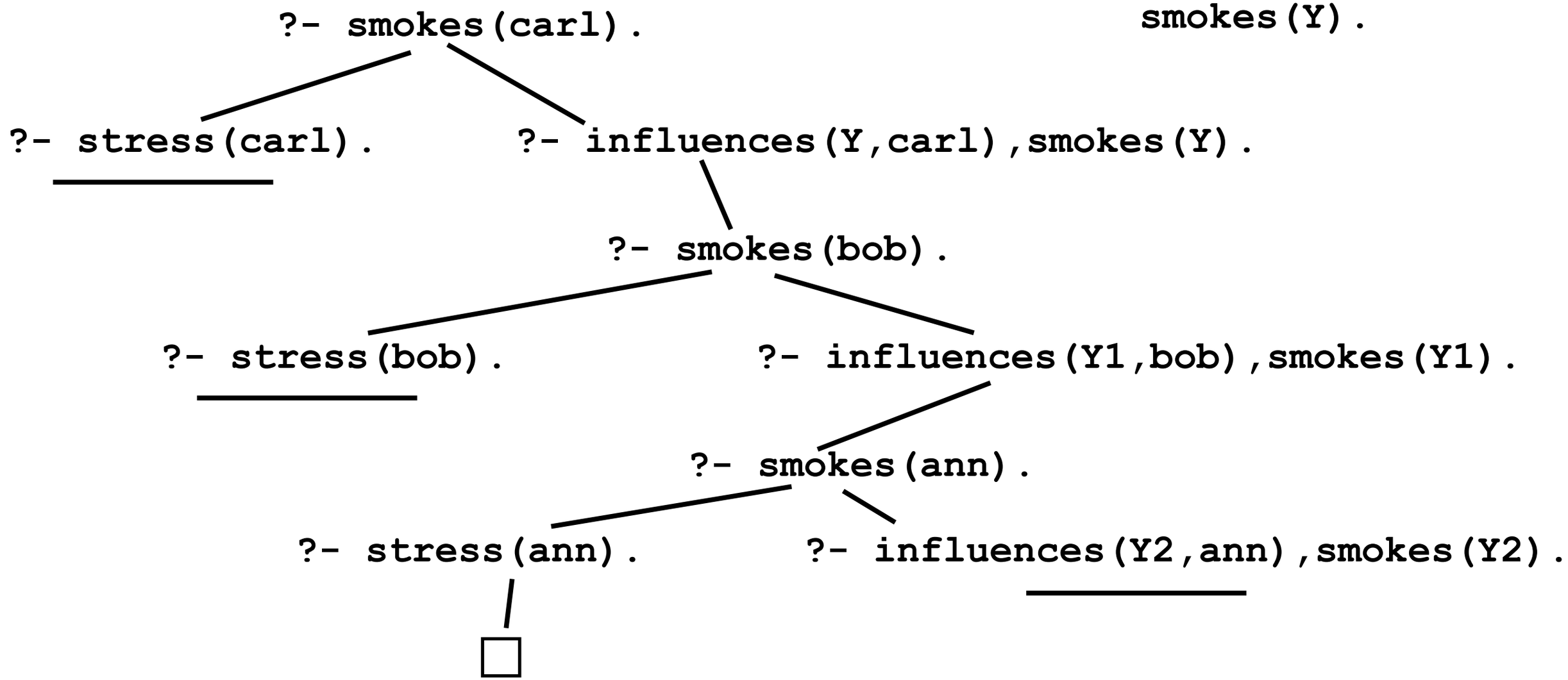
Logical Reasoning: Proofs in Prolog

```
stress(ann).
influences(ann,bob).
influences(bob,carl).

smokes(X)  :- stress(X).
smokes(X)  :-
      influences(Y,X),
      smokes(Y).
```



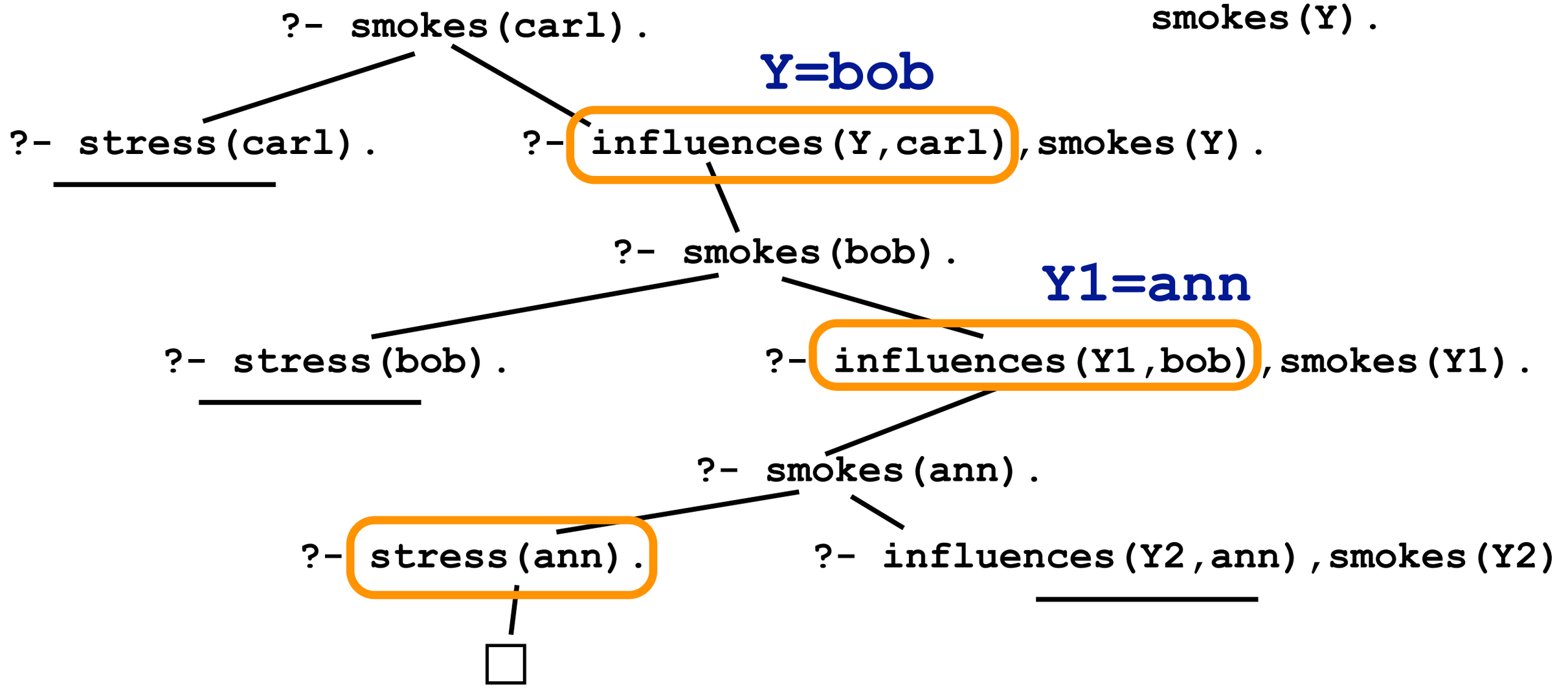
Logical Reasoning: Proofs in Prolog

```
stress(ann).
influences(ann,bob).
influences(bob,carl).

smokes(X)  :- stress(X).
smokes(X)  :-
      influences(Y,X),
      smokes(Y).
```



Logical Reasoning: Proofs in Prolog

```
stress(ann).
influences(ann,bob).
influences(bob,carl).

smokes(X) :- stress(X).
smokes(X) :-
      influences(Y,X),
      smokes(Y).
```

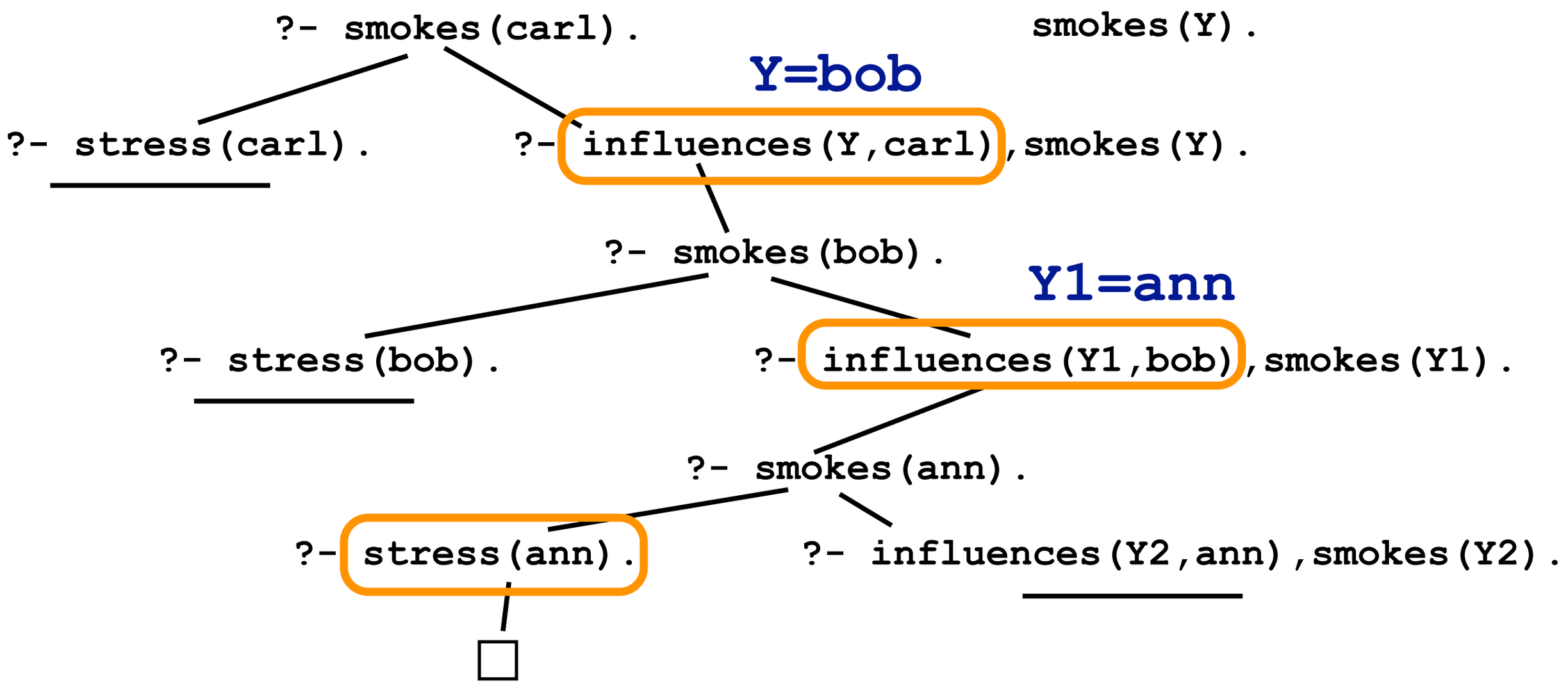


proof = facts used in successful derivation:

`influences(bob,carl) &influences(ann,bob) &stress(ann)`

Proofs in ProbLog

```
0.8::stress(ann).
0.6::influences(ann,bob).
0.2::influences(bob,carl).

smokes(X) :- stress(X).
smokes(X) :-
     influences(Y,X),
     smokes(Y).
```

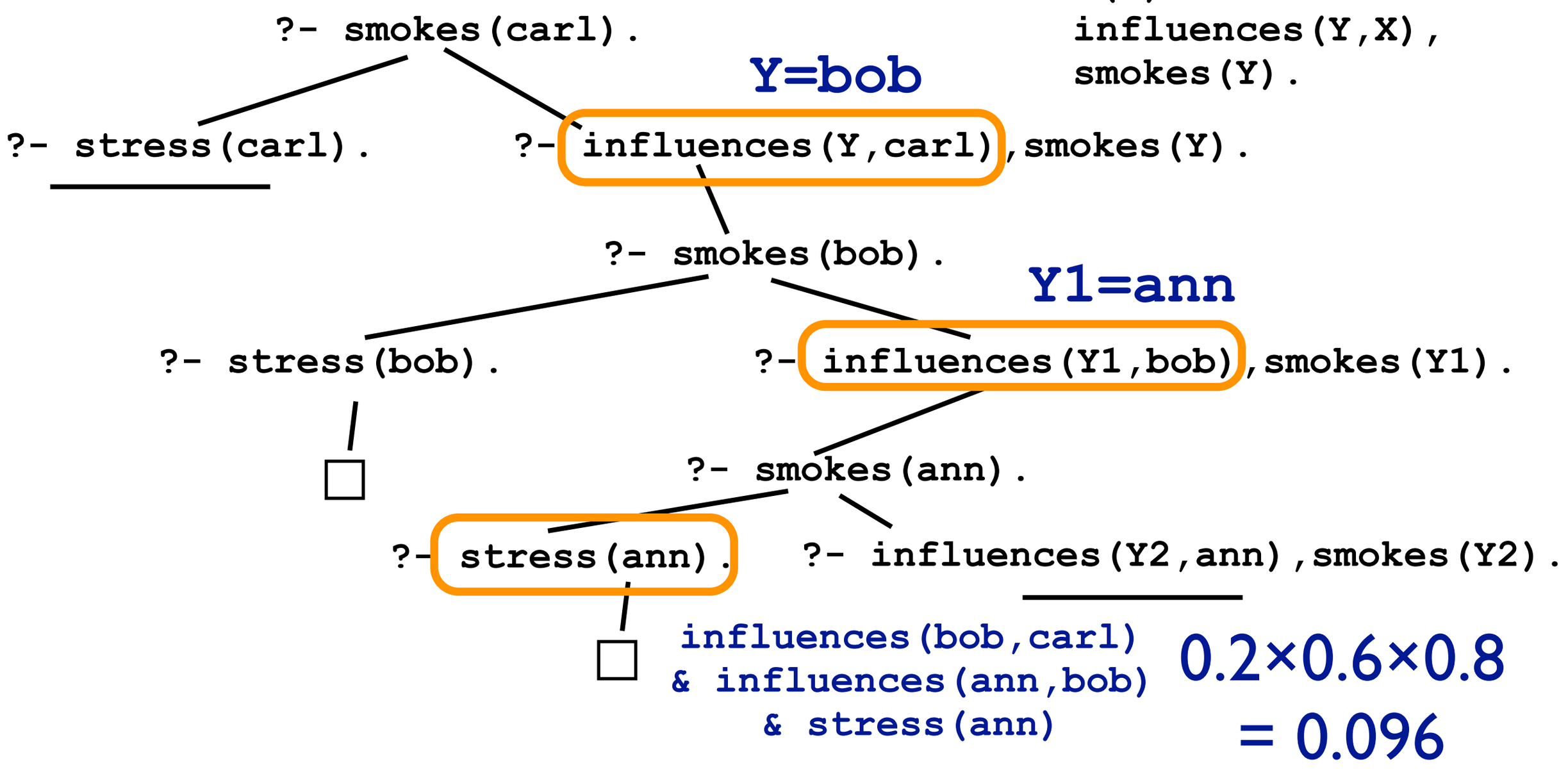


`influences(bob,carl)&influences(ann,bob)&stress(ann)`

probability of proof = 0.2 × 0.6 × 0.8 = 0.096

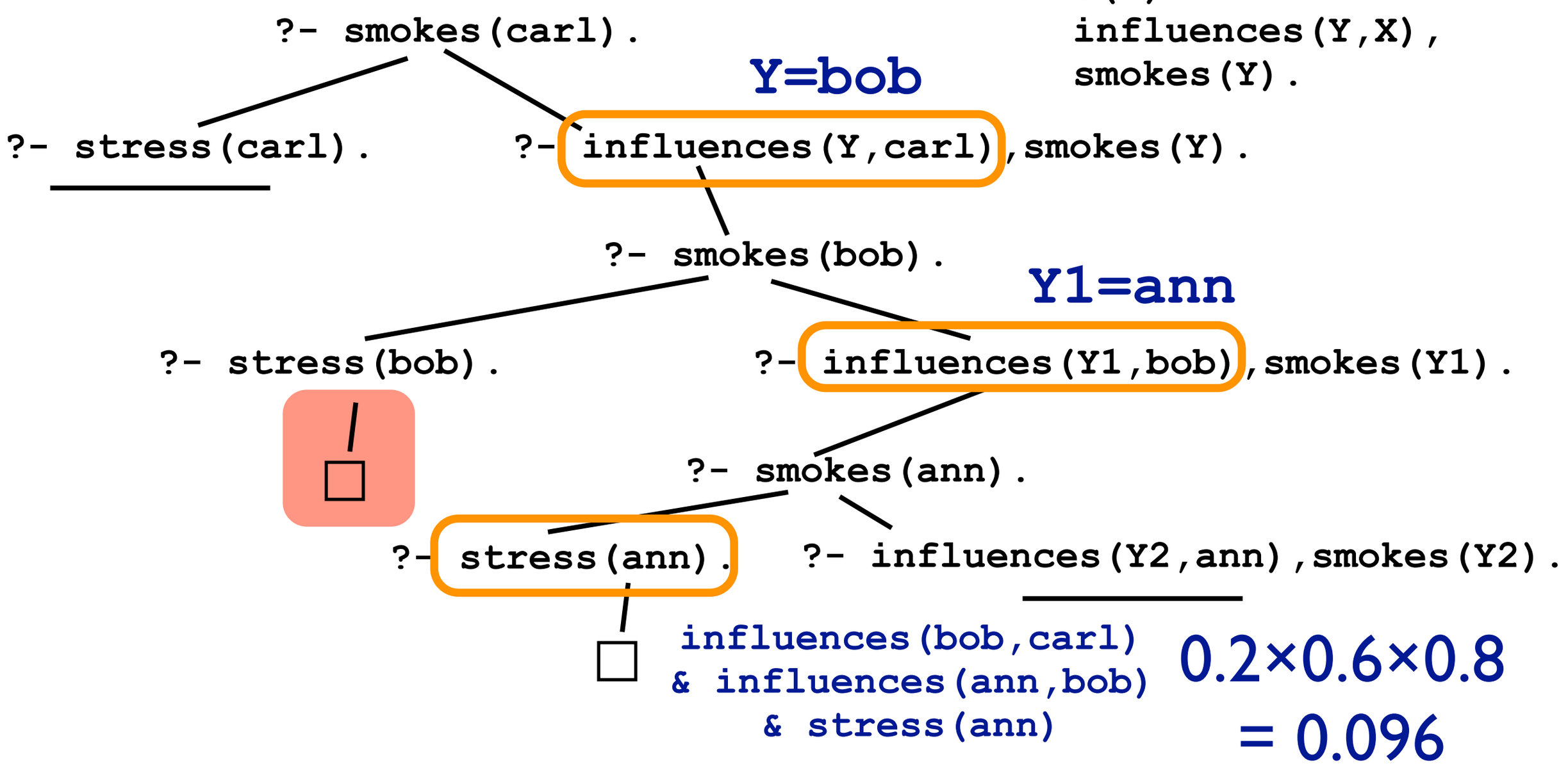
Proofs in ProbLog

```
0.8::stress(ann).
0.4::stress(bob).
0.6::influences(ann,bob).
0.2::influences(bob,carl).

smokes(X) :- stress(X).
smokes(X) :-
     influences(Y,X),
     smokes(Y).
```



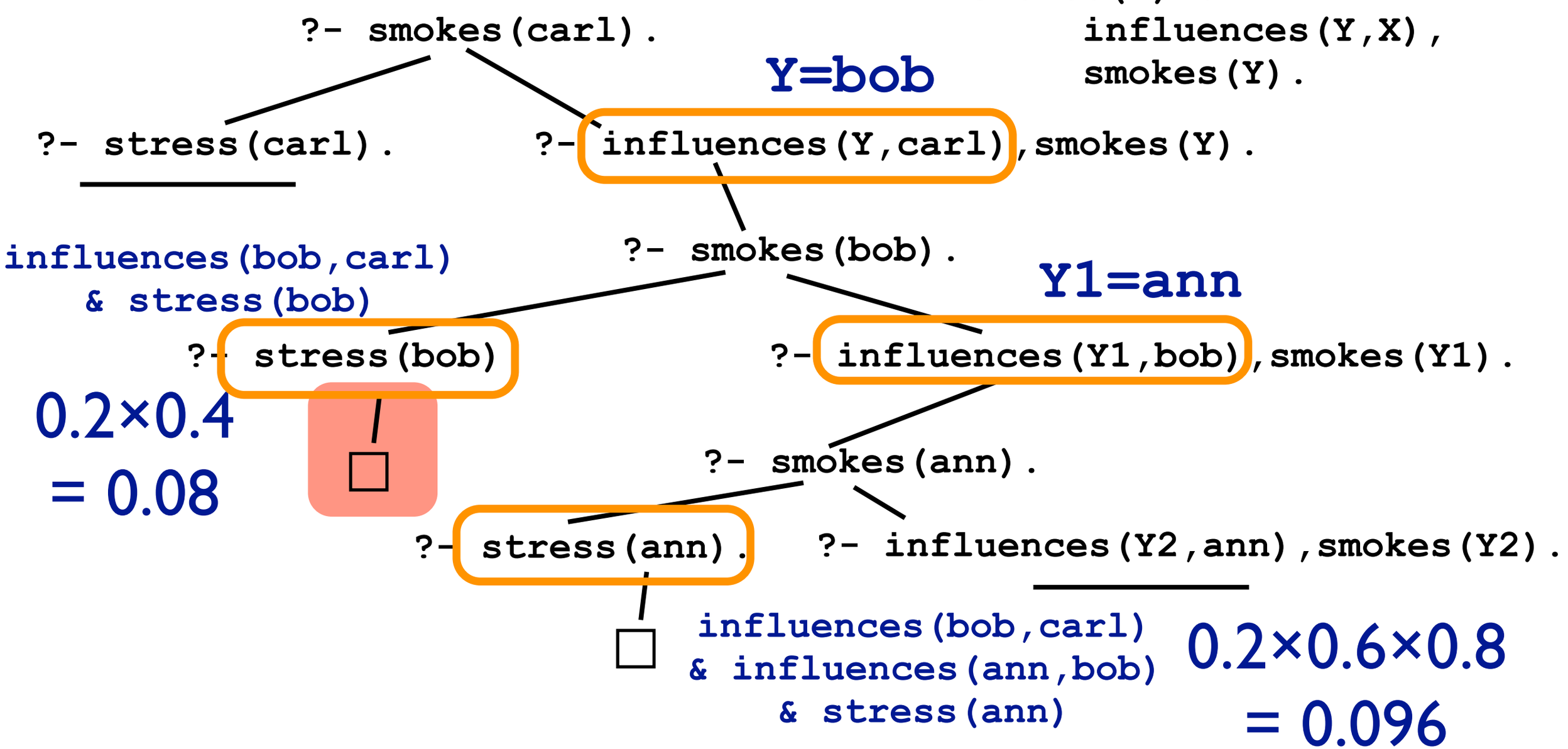
Proofs in ProbLog

```
0.8::stress(ann).
0.4::stress(bob).
0.6::influences(ann,bob).
0.2::influences(bob,carl).

smokes(X) :- stress(X).
smokes(X) :-
     influences(Y,X),
     smokes(Y).
```



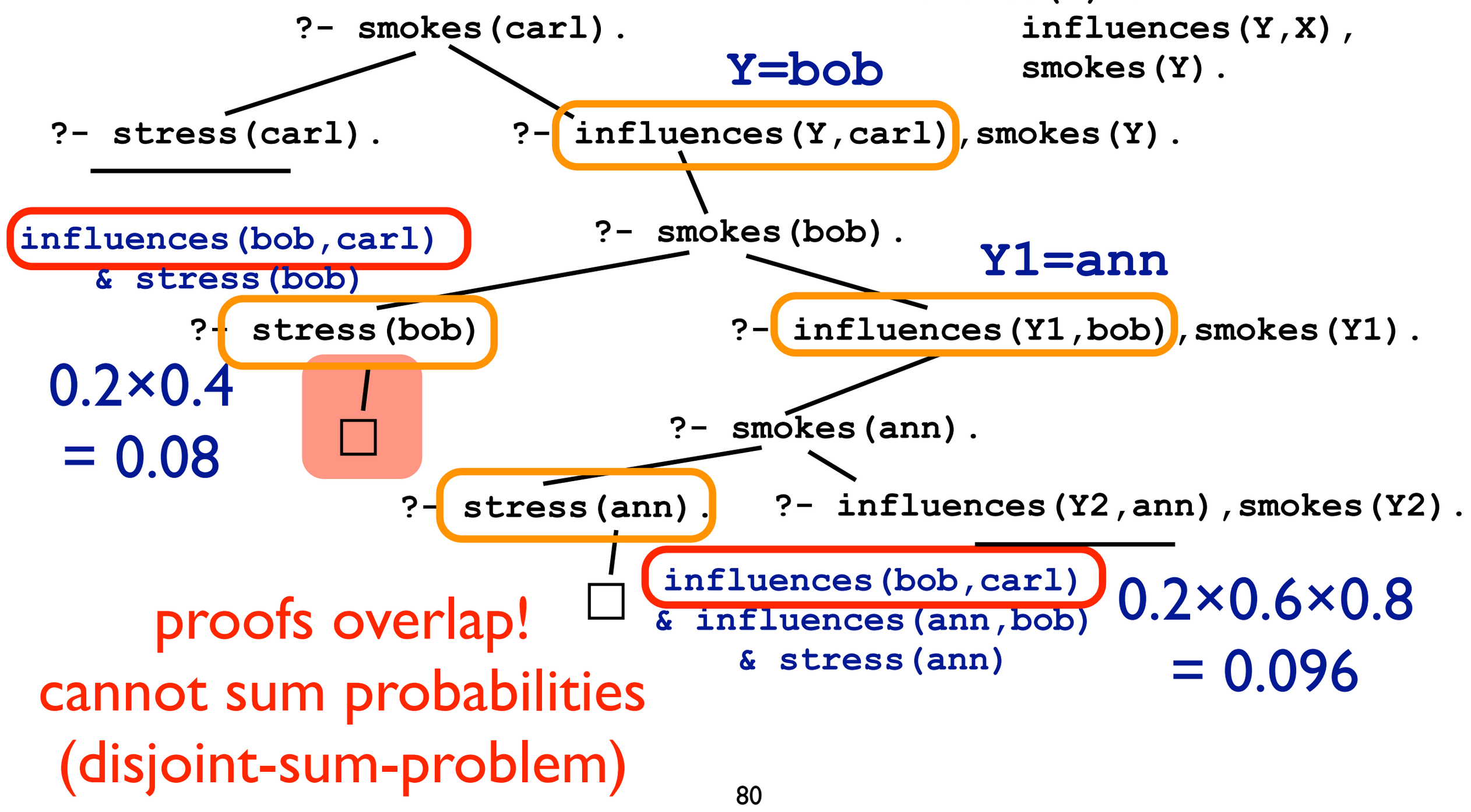
Proofs in ProbLog

```
0.8::stress(ann).
0.4::stress(bob).
0.6::influences(ann,bob).
0.2::influences(bob,carl).

smokes(X) :- stress(X).
smokes(X) :-
     influences(Y,X),
     smokes(Y).
```



Proofs in ProbLog

```
0.8::stress(ann).
0.4::stress(bob).
0.6::influences(ann,bob).
0.2::influences(bob,carl).

smokes(X) :- stress(X).
smokes(X) :-
     influences(Y,X),
     smokes(Y).
```



proofs overlap!
cannot sum probabilities
(disjoint-sum-problem)

Disjoint-Sum-Problem

possible worlds

```
infl(bob,carl) &   infl(ann,bob) &   st(ann) & \+st(bob)

infl(bob,carl) &   infl(ann,bob) &   st(ann) &   st(bob)

infl(bob,carl) & \+infl(ann,bob) &   st(ann) &   st(bob)

infl(bob,carl) &   infl(ann,bob) & \+st(ann) &   st(bob)

infl(bob,carl) & \+infl(ann,bob) & \+st(ann) &   st(bob)

...
```

Disjoint-Sum-Problem

possible worlds

influences(bob,carl) &
influences(ann,bob) & stress(ann)

```
infl(bob,carl) &   infl(ann,bob) &   st(ann) & \+st(bob)

infl(bob,carl) &   infl(ann,bob) &   st(ann) &   st(bob)

infl(bob,carl) & \+infl(ann,bob) &   st(ann) &   st(bob)

infl(bob,carl) &   infl(ann,bob) & \+st(ann) &   st(bob)

infl(bob,carl) & \+infl(ann,bob) & \+st(ann) &   st(bob)

...
```

Disjoint-Sum-Problem

possible worlds

influences(bob,carl) &
influences(ann,bob) & stress(ann)

```
infl(bob,carl) &   infl(ann,bob) &   st(ann) & \+st(bob)
infl(bob,carl) &   infl(ann,bob) &   st(ann) &   st(bob)
infl(bob,carl) & \+infl(ann,bob) &   st(ann) &   st(bob)
infl(bob,carl) &   infl(ann,bob) & \+st(ann) &   st(bob)
infl(bob,carl) & \+infl(ann,bob) & \+st(ann) &   st(bob)
```

... **influences(bob,carl) & stress(bob)**

Disjoint-Sum-Problem

possible worlds

influences(bob,carl) &
influences(ann,bob) & stress(ann)

```
infl(bob,carl) &   infl(ann,bob) &   st(ann) & \+st(bob)
infl(bob,carl) &   infl(ann,bob) &   st(ann) &   st(bob)
infl(bob,carl) & \+infl(ann,bob) &   st(ann) &   st(bob)
infl(bob,carl) &   infl(ann,bob) & \+st(ann) &   st(bob)
infl(bob,carl) & \+infl(ann,bob) & \+st(ann) &   st(bob)
```

... **influences(bob,carl) & stress(bob)**

sum of proof probabilities: 0.096+0.08 = 0.1760

Disjoint-Sum-Problem

possible worlds

influences(bob,carl) &
influences(ann,bob) & stress(ann)

infl(bob,carl) & infl(ann,bob) & st(ann) & \+st(bob)	0.0576
infl(bob,carl) & infl(ann,bob) & st(ann) & st(bob)	0.0384
infl(bob,carl) & \+infl(ann,bob) & st(ann) & st(bob)	0.0256
infl(bob,carl) & infl(ann,bob) & \+st(ann) & st(bob)	0.0096
infl(bob,carl) & \+infl(ann,bob) & \+st(ann) & st(bob)	0.0064
... **influences(bob,carl) & stress(bob)**	∑ = 0.1376

sum of proof probabilities: 0.096+0.08 = 0.1760

Disjoint-Sum-Problem

possible worlds

solution: knowledge compilation

`infl(bob,carl) & infl(ann,bob) & st(ann) & \+st(bob)`	0.0576
`infl(bob,carl) & infl(ann,bob) & st(ann) & st(bob)`	0.0384
`infl(bob,carl) & \+infl(ann,bob) & st(ann) & st(bob)`	0.0256
`infl(bob,carl) & infl(ann,bob) & \+st(ann) & st(bob)`	0.0096
`infl(bob,carl) & \+infl(ann,bob) & \+st(ann) & st(bob)`	0.0064
`...` `influences(bob,carl) & stress(bob)`	$\sum = 0.1376$

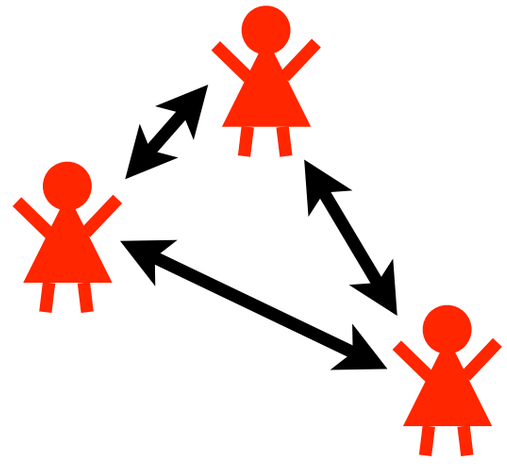
sum of proof probabilities: 0.096+0.08 = 0.1760

Complexity of Querying

Complexity of querying in probabilistic databases

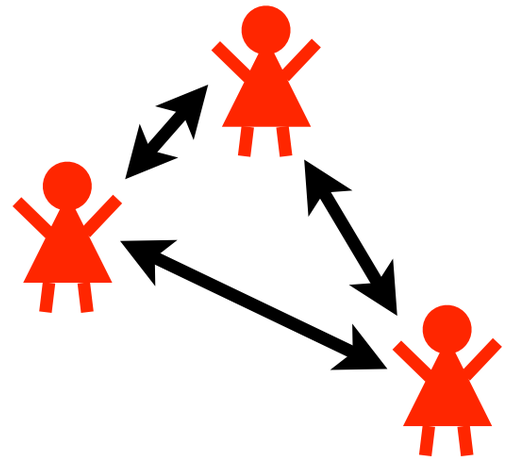
- queries have fixed size (no recursion)
- size of query << size of database
- complexity of evaluating given query measured in size of database (data complexity)
- standard relational database queries: polynomial

How hard is evaluating q?



```
q1 :- stress(X), influences(X,Y).

0.7::stress(1).
0.4::stress(2).
0.9::stress(3).

0.83::influences(1,2).
0.41::influences(1,3).
...
0.17::influences(3,2).
```

How hard is evaluating q?



```
q1 :- stress(X), influences(X,Y).
```

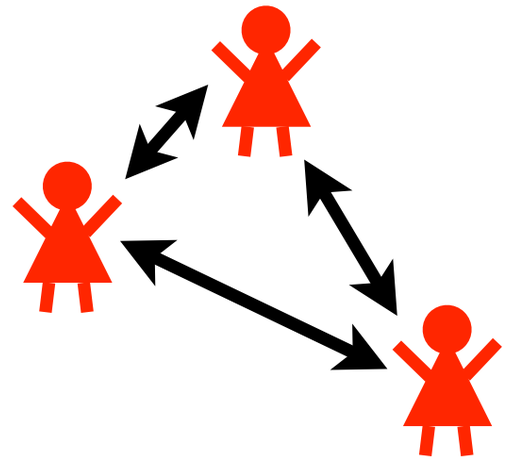
```
0.7::stress(1).
0.4::stress(2).
0.9::stress(3).
```

```
0.83::influences(1,2).
0.41::influences(1,3).
...
0.17::influences(3,2).
```

proofs

```
s(1),i(1,2)
s(1),i(1,3)
s(2),i(2,1)
s(2),i(2,3)
s(3),i(3,1)
s(3),i(3,2)
```

How hard is evaluating q?



```
q1 :- stress(X), influences(X,Y).
```

```
0.7::stress(1).
0.4::stress(2).
0.9::stress(3).
```

```
0.83::influences(1,2).
0.41::influences(1,3).
...
0.17::influences(3,2).
```

proofs

`s(1)`,`i(1,2)`

`s(1)`,`i(1,3)`

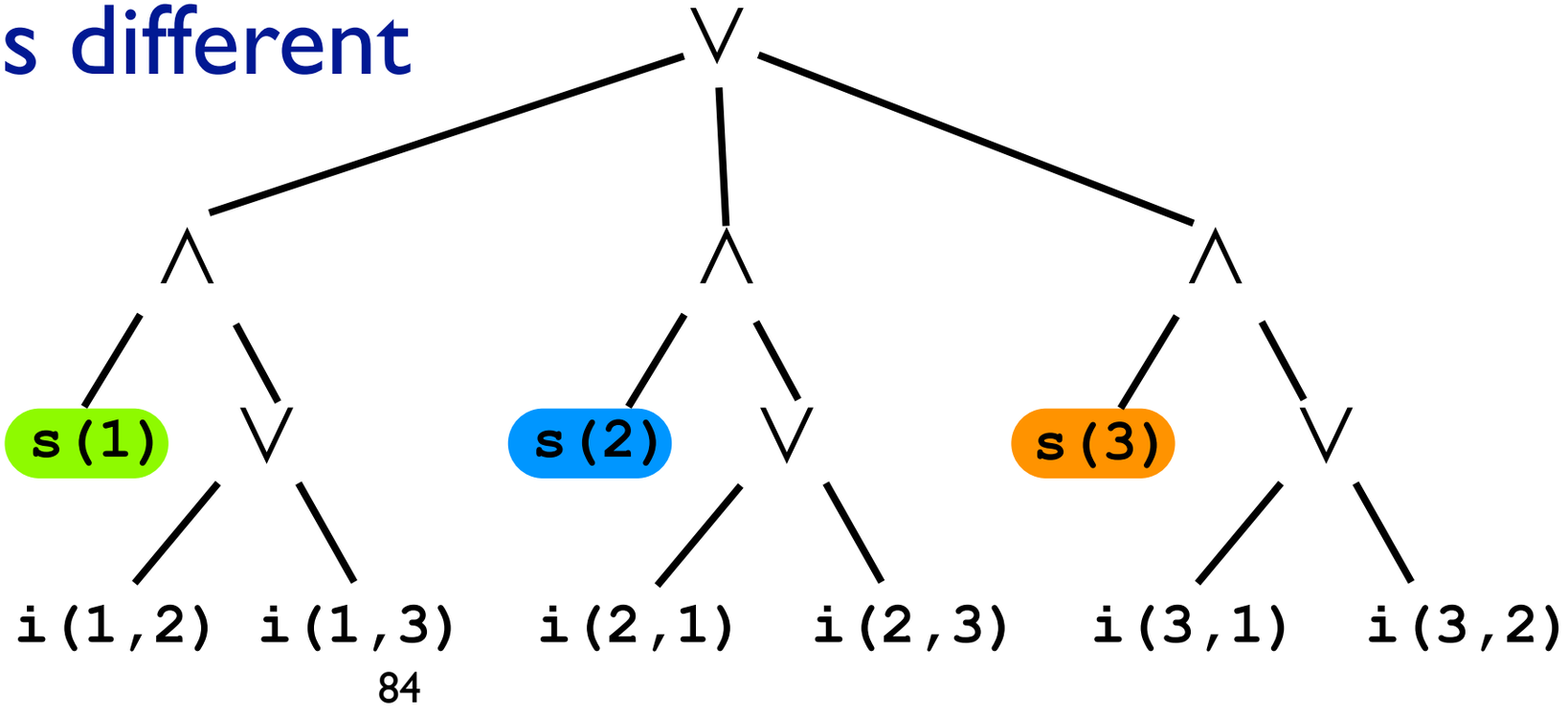
`s(2)`,`i(2,1)`

`s(2)`,`i(2,3)`

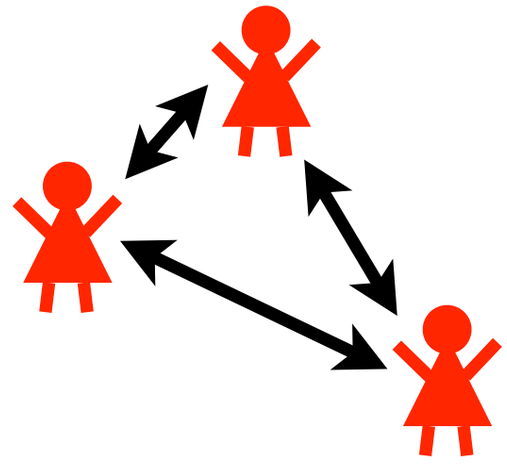
`s(3)`,`i(3,1)`

`s(3)`,`i(3,2)`

tree structure, all leaves different



How hard is evaluating q?



```
q1 :- stress(X), influences(X,Y).
```

```
0.7::stress(1).
0.4::stress(2).
0.9::stress(3).
```

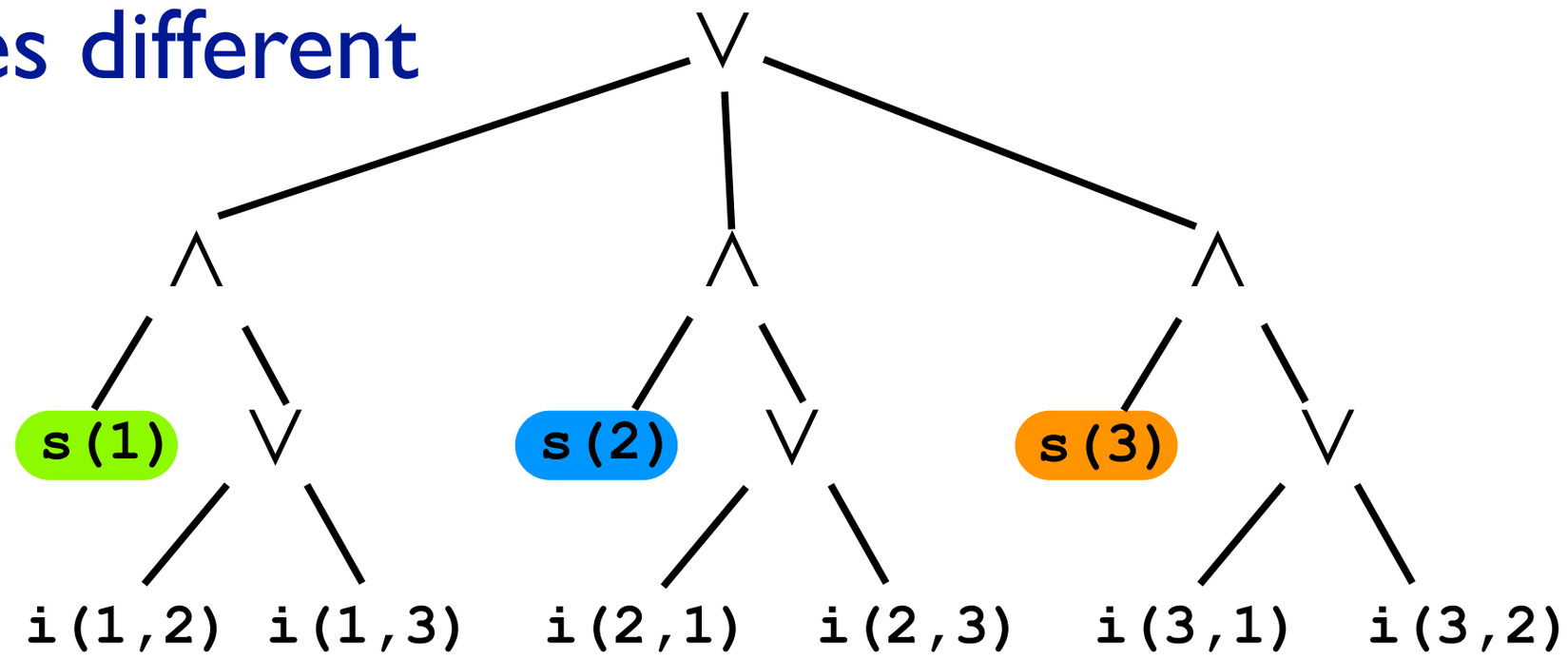
```
0.83::influences(1,2).
0.41::influences(1,3).
...
0.17::influences(3,2).
```

proofs

- s(1),i(1,2)
- s(1),i(1,3)
- s(2),i(2,1)
- s(2),i(2,3)
- s(3),i(3,1)
- s(3),i(3,2)

tree structure, all leaves different

$$P(\varphi_1 \wedge \varphi_2) = P(\varphi_1) \cdot P(\varphi_2)$$
$$P(\varphi_1 \vee \varphi_2) = 1 - (1 - P(\varphi_1)) \cdot (1 - P(\varphi_2))$$



$$P(\mathtt{q}) = 1 - \prod_{j=1..n} (1 - \left(P(\mathtt{s}(\mathtt{X_j}))(1 - \prod_{k=1..n, k \neq j} (1 - P(\mathtt{i}(\mathtt{X_j}, \mathtt{Y_k})))) \right))$$

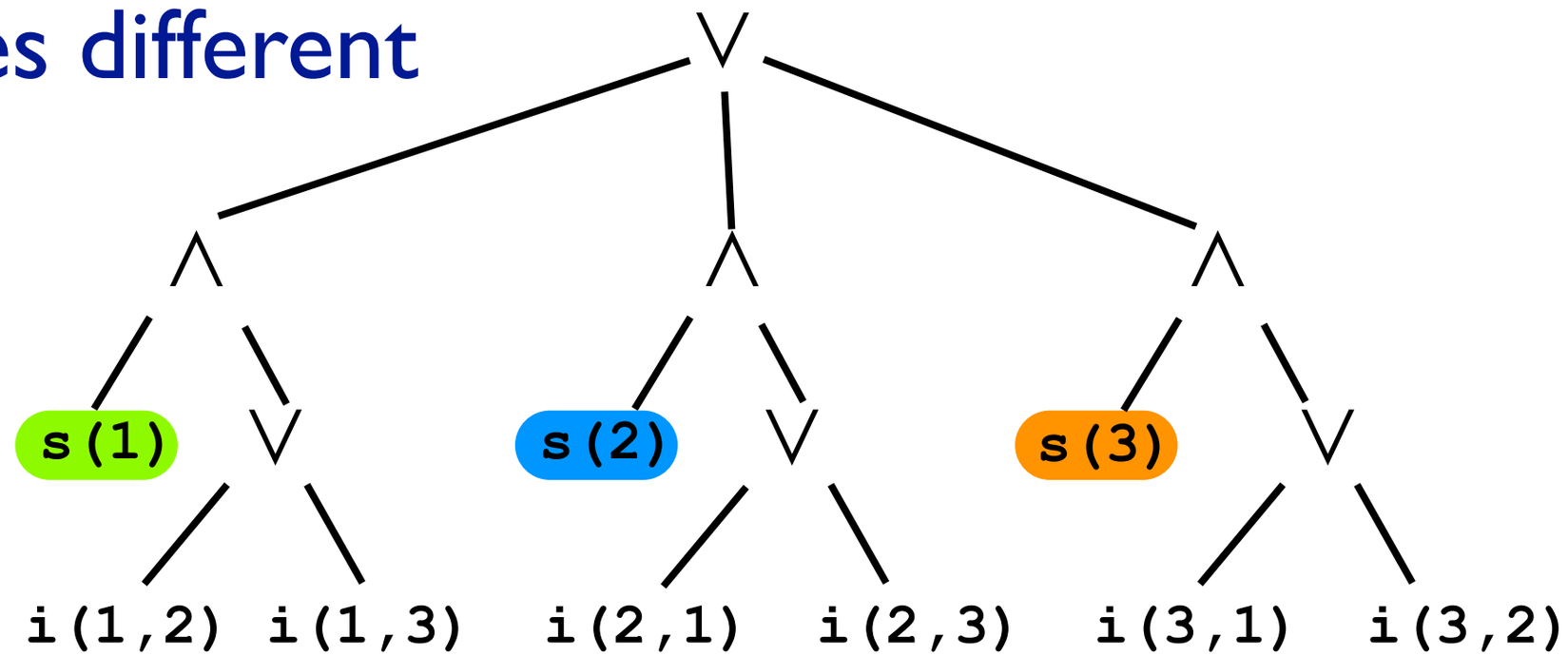
polynomial in database size / number of persons n

proofs

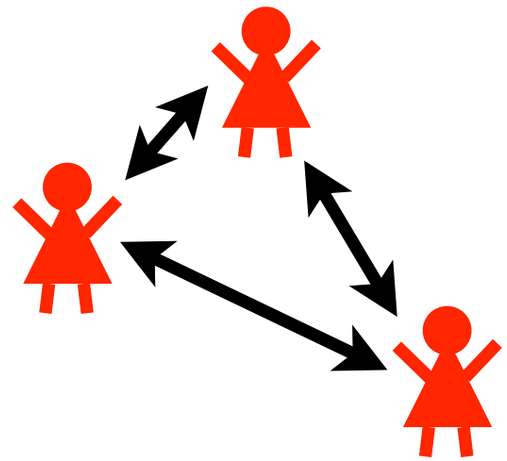
- s(1),i(1,2)
- s(1),i(1,3)
- s(2),i(2,1)
- s(2),i(2,3)
- s(3),i(3,1)
- s(3),i(3,2)

tree structure, all leaves different

$$P(\varphi_1 \wedge \varphi_2) = P(\varphi_1) \cdot P(\varphi_2)$$
$$P(\varphi_1 \vee \varphi_2) = 1 - (1 - P(\varphi_1)) \cdot (1 - P(\varphi_2))$$



How hard is evaluating q?



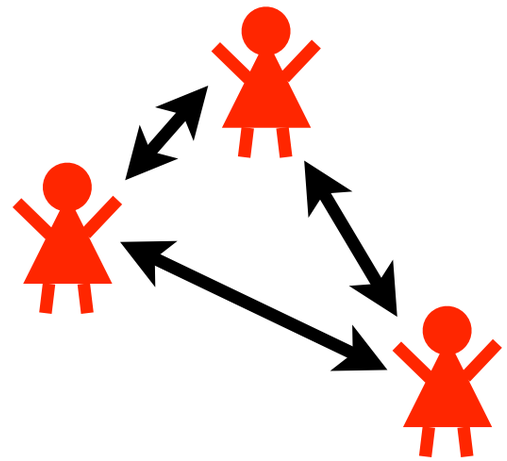
```
q2 :- stress(X), influences(X,Y), male(Y).
```

```
0.3::male(1).
0.8::male(2).
0.9::male(3).
```

```
0.7::stress(1).
0.4::stress(2).
0.9::stress(3).
```

```
0.83::influences(1,2).
0.41::influences(1,3).
...
0.17::influences(3,2).
```

How hard is evaluating q?



```
q2 :- stress(X), influences(X,Y), male(Y).
```

```
0.3::male(1).
0.8::male(2).
0.9::male(3).
```

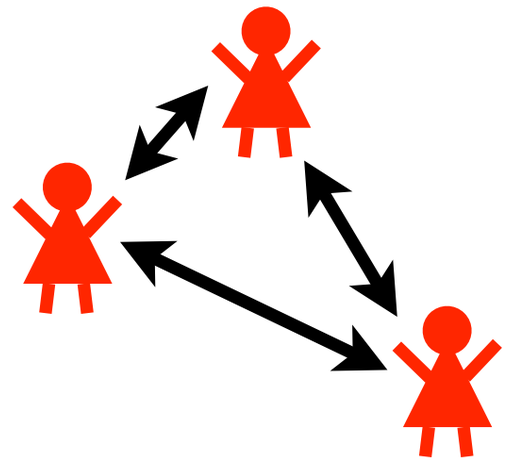
```
0.7::stress(1).
0.4::stress(2).
0.9::stress(3).
```

```
0.83::influences(1,2).
0.41::influences(1,3).
...
0.17::influences(3,2).
```

proofs

```
s(1),i(1,2),m(2)
s(1),i(1,3),m(3)
s(2),i(2,1),m(1)
s(2),i(2,3),m(3)
s(3),i(3,1),m(1)
s(3),i(3,2),m(2)
```

How hard is evaluating q?



```
q2 :- stress(X), influences(X,Y), male(Y).
```

```
0.3::male(1).
0.8::male(2).
0.9::male(3).
```

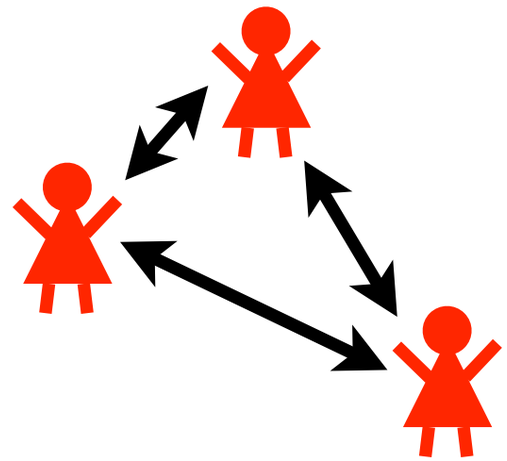
```
0.7::stress(1).
0.4::stress(2).
0.9::stress(3).
```

```
0.83::influences(1,2).
0.41::influences(1,3).
...
0.17::influences(3,2).
```

proofs

```
s(1),i(1,2),m(2)
s(1),i(1,3),m(3)
s(2),i(2,1),m(1)
s(2),i(2,3),m(3)
s(3),i(3,1),m(1)
s(3),i(3,2),m(2)
```

How hard is evaluating q?



```
q2 :- stress(X), influences(X,Y), male(Y).
```

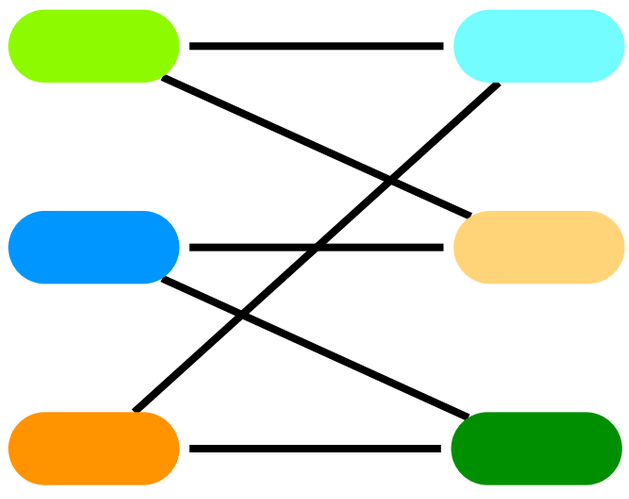
```
0.3::male(1).
0.8::male(2).
0.9::male(3).
```

```
0.7::stress(1).
0.4::stress(2).
0.9::stress(3).
```

```
0.83::influences(1,2).
0.41::influences(1,3).
...
0.17::influences(3,2).
```

proofs

```
s(1),i(1,2),m(2)
s(1),i(1,3),m(3)
s(2),i(2,1),m(1)
s(2),i(2,3),m(3)
s(3),i(3,1),m(1)
s(3),i(3,2),m(2)
```



cannot build tree structure with all leaves different

Read-Once Formulas

- Propositional formulas that can be rewritten such that each variable occurs at most once (= as tree with all leaves different)

Read-Once Formulas

- Propositional formulas that can be rewritten such that each variable occurs at most once (= as tree with all leaves different)
- Can be evaluated in polynomial time

Read-Once Formulas

- Propositional formulas that can be rewritten such that each variable occurs at most once (= as tree with all leaves different)
- Can be evaluated in polynomial time
- Unate formula: read once ↔ P4-free & normal

Read-Once Formulas

- Propositional formulas that can be rewritten such that each variable occurs at most once (= as tree with all leaves different)
- Can be evaluated in polynomial time
- Unate formula: read once ↔ P4-free & normal

no variable appears both pos & neg

X∨Y is unate

(X∨Y) ∧ (¬X∨Z) not

Dichotomy of UCQ Evaluation

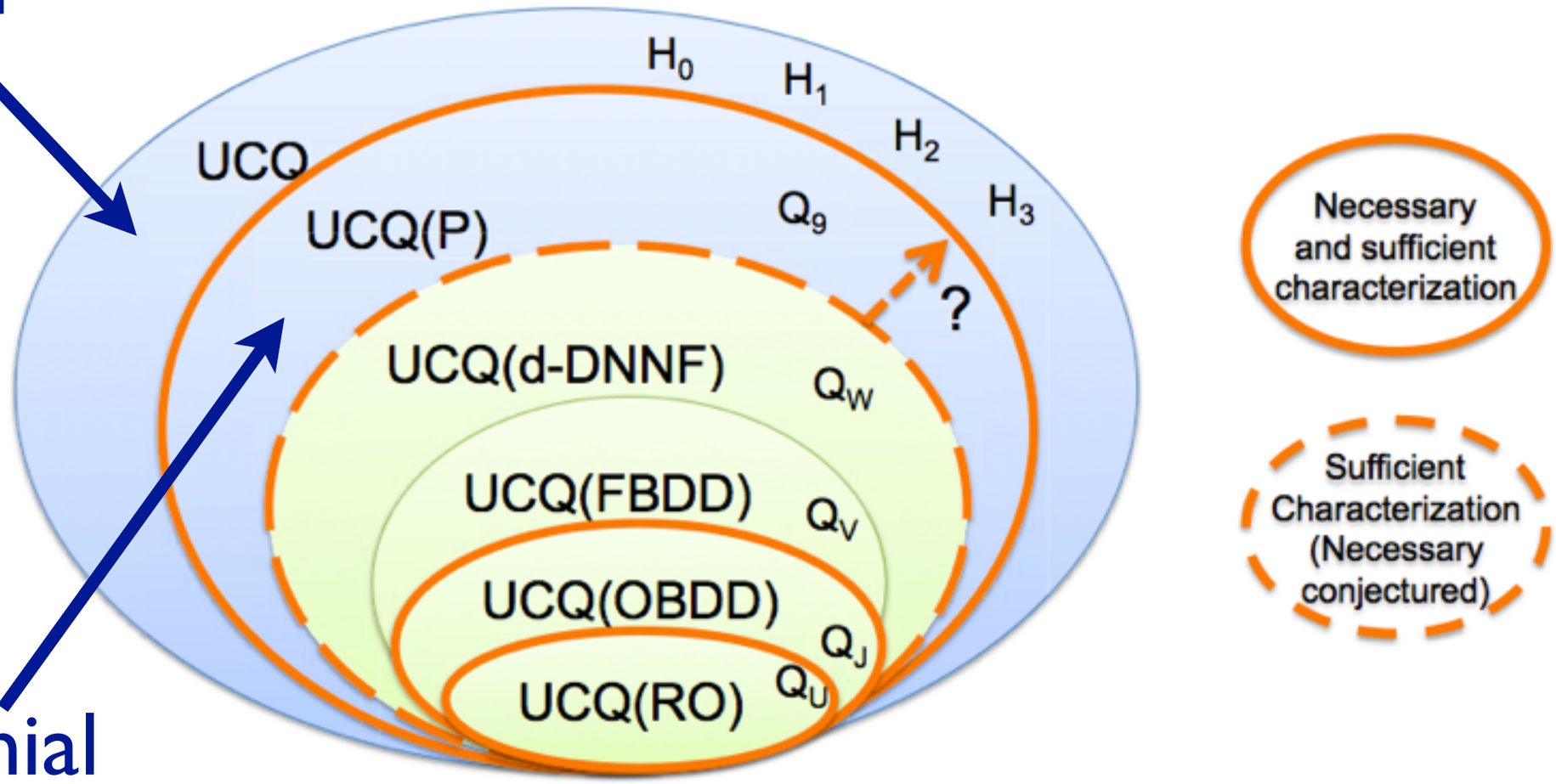
- Union of Conjunctive Queries
  ≈ Datalog without recursion and negation
- Theorem: UCQ evaluation is either polynomial in database size or #P-hard

Dichotomy of UCQ Evaluation

- Union of Conjunctive Queries
  ≈ Datalog without recursion and negation
- Theorem: UCQ evaluation is either polynomial in database size or #P-hard

counting version of NP decision problems, e.g., model counting

#P-hard

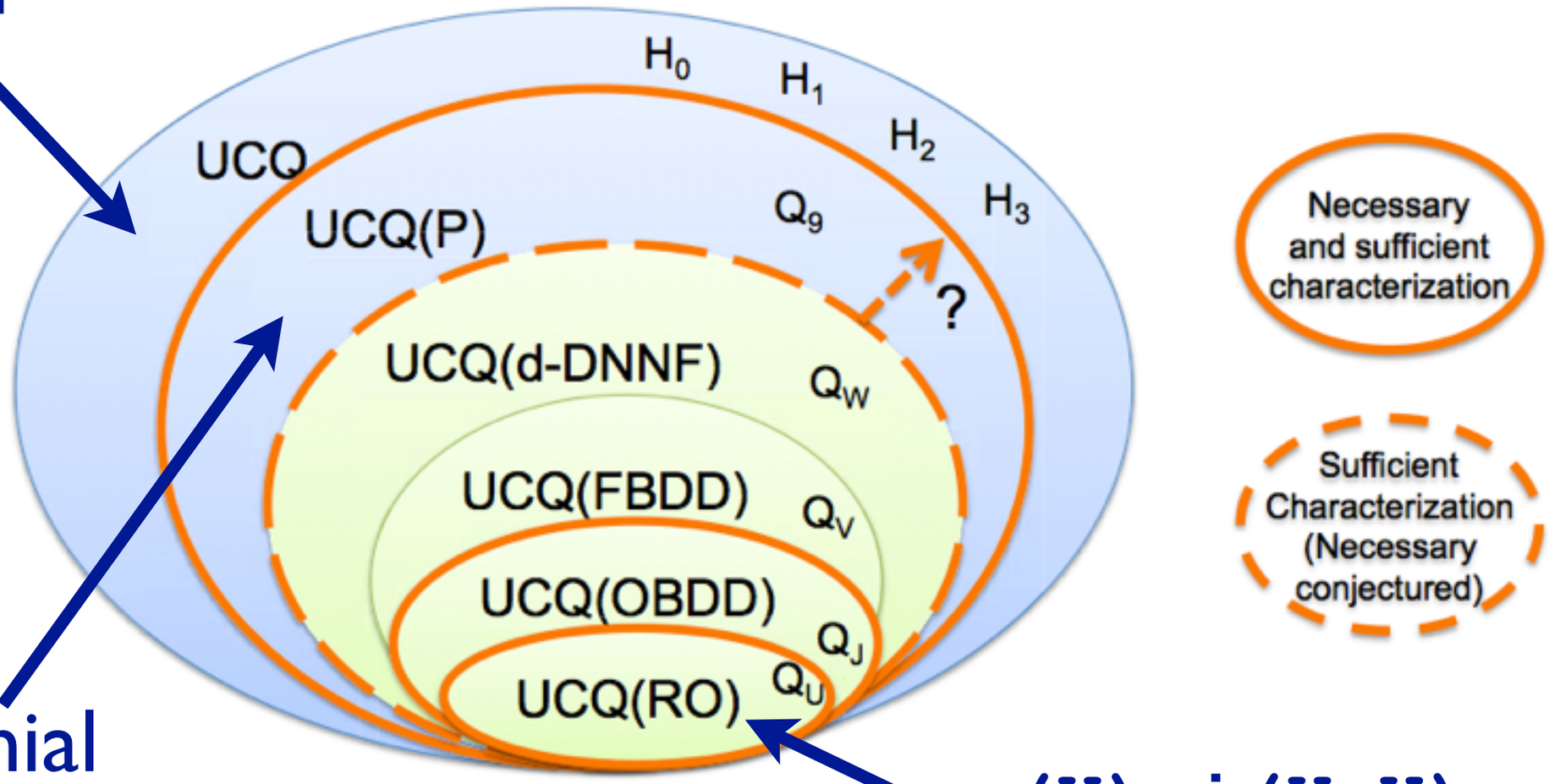


polynomial

Figure 5.4: The query compilation hierarchy for Unions of Conjunctive Queries (UCQ).

Fig. from [Suciu et al 2011]

#P-hard



polynomial

s(X),i(X,Y)

Figure 5.4: The query compilation hierarchy for Unions of Conjunctive Queries (UCQ).

Fig. from [Suciu et al 2011]

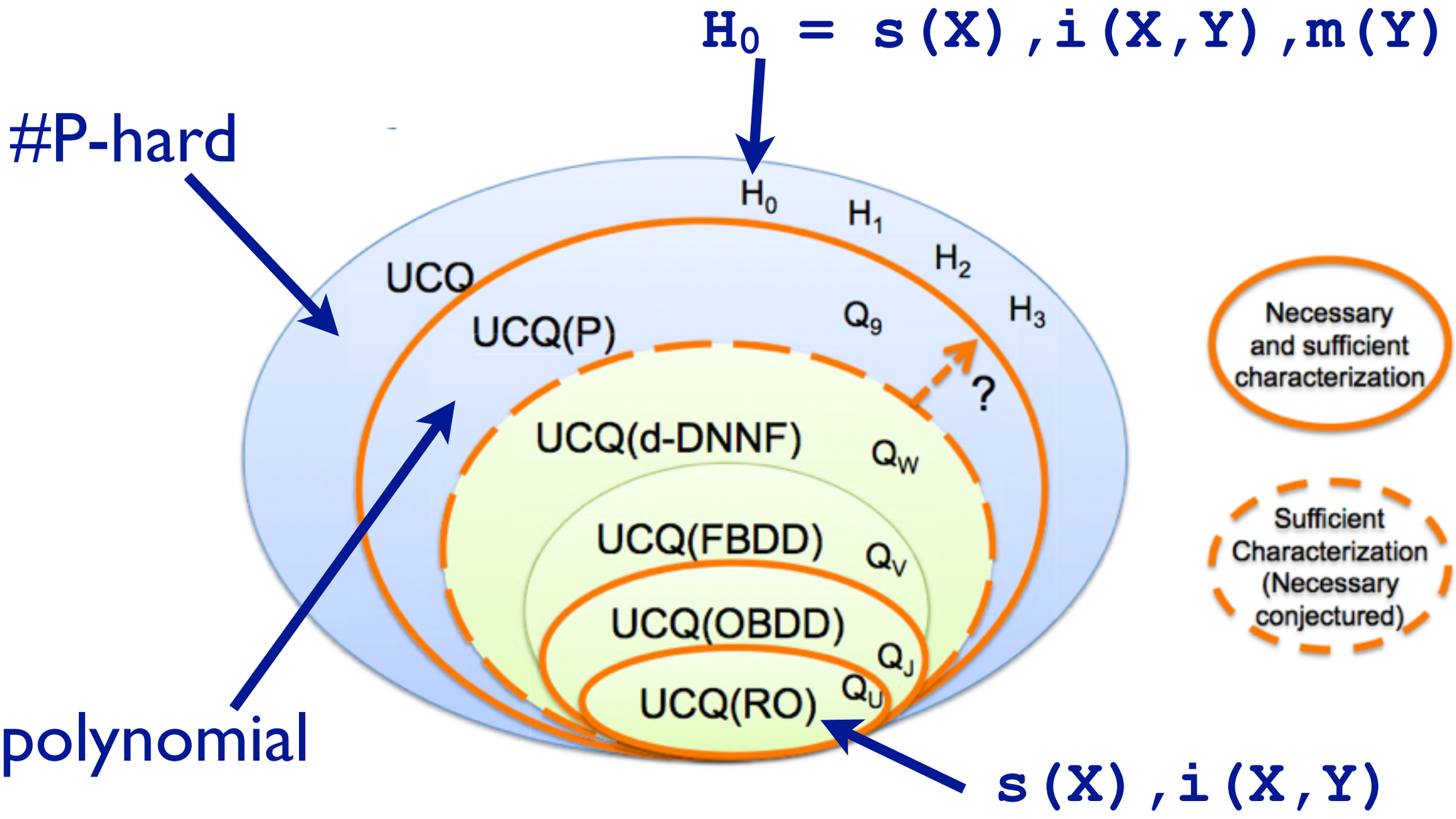


Figure 5.4: The query compilation hierarchy for Unions of Conjunctive Queries (UCQ).

Fig. from [Suciu et al 2011]

Two Steps

- **Logical inference -**
  - about a ground logical theory
    - proofs or model theoretic …
      - *Result: Weighted Model Counting problem*
- **Probabilistic propositional inference** —
  - Backtracking search — DPLL, VE, RC based
  - Knowledge Compilation
- **Advanced** — lifted inference

Binary Decision Diagrams [Bryant 86]

- compact graphical representation of Boolean formula
- popular in many branches of CS
- automatically disjoins proofs
  → efficient probability computation

Binary Decision Diagrams
[Bryant 86]

$X \vee Y \vee Z$

Binary Decision Diagrams
[Bryant 86]

$X \vee Y \vee Z$

X	0	0	0	0	1	1	1	1
Y	0	0	1	1	0	0	1	1
Z	0	1	0	1	0	1	0	1

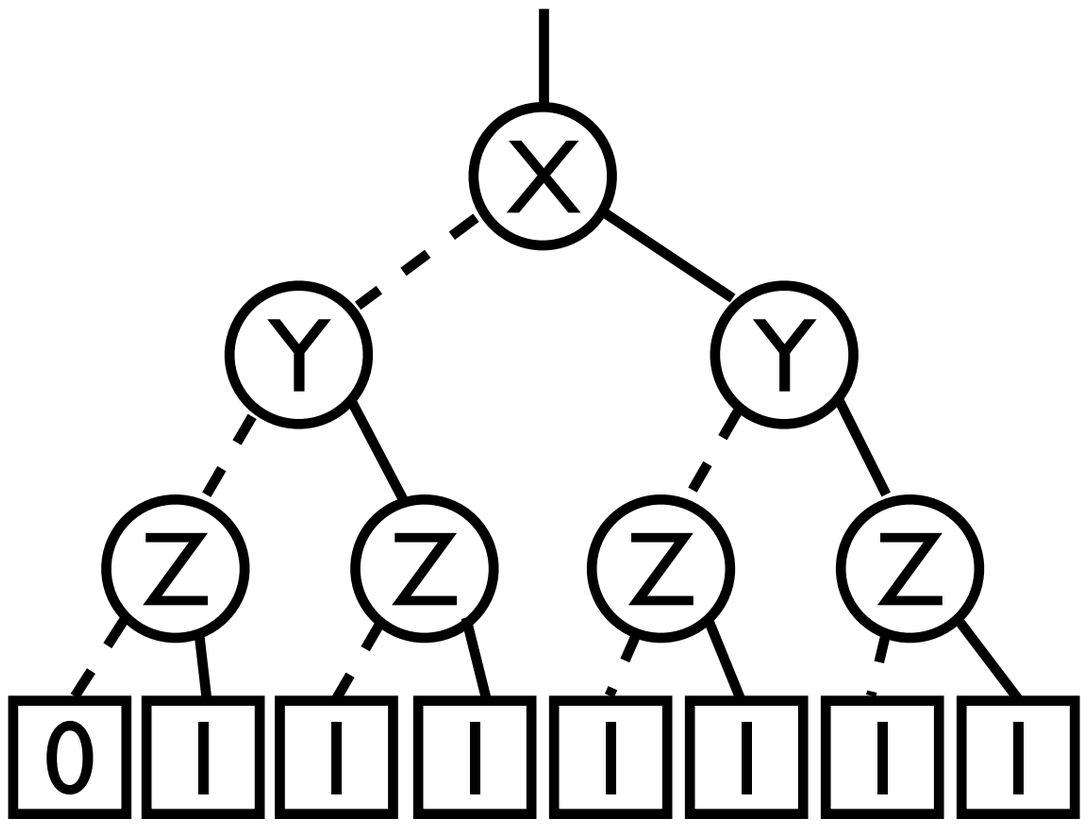
Binary Decision Diagrams
[Bryant 86]

$X \vee Y \vee Z$

X	0	0	0	0	1	1	1	1
Y	0	0	1	1	0	0	1	1
Z	0	1	0	1	0	1	0	1

Binary Decision Diagrams

[Bryant 86]

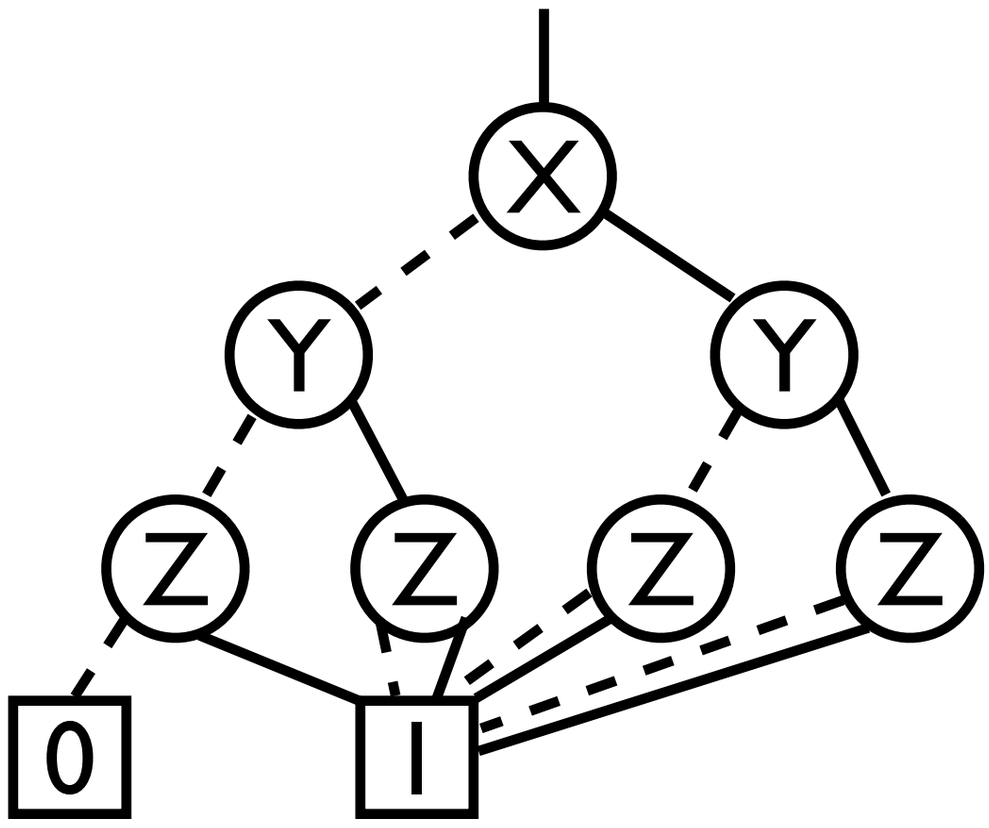


$X \vee Y \vee Z$

X	0	0	0	0	1	1	1	1
Y	0	0	1	1	0	0	1	1
Z	0	1	0	1	0	1	0	1

Binary Decision Diagrams

[Bryant 86]

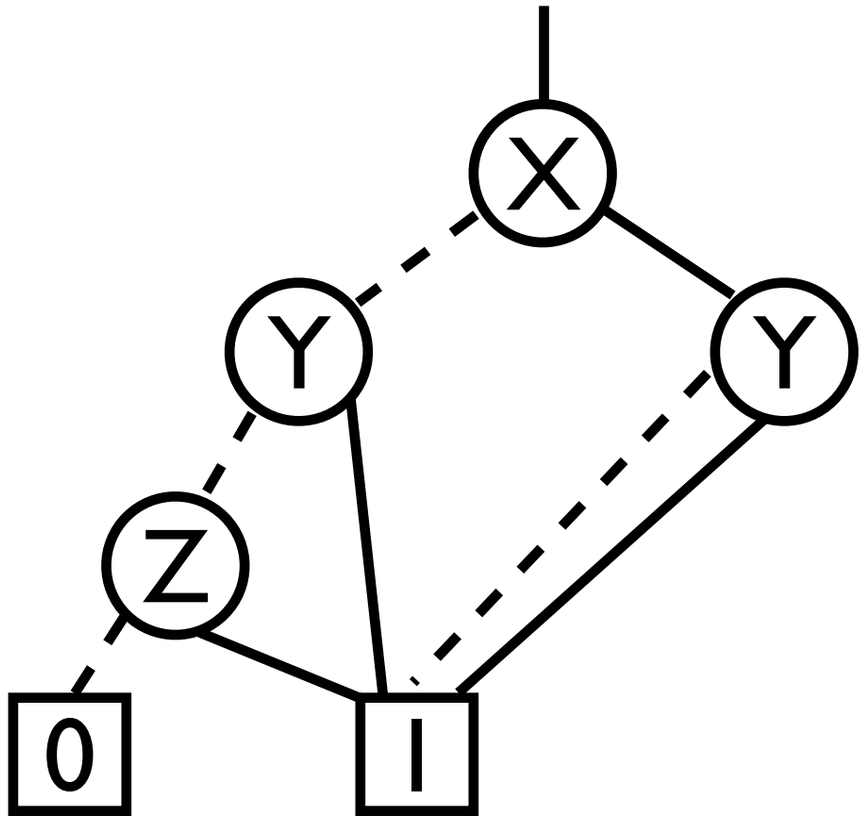


$X \vee Y \vee Z$

X	0	0	0	0	1	1	1	1
Y	0	0	1	1	0	0	1	1
Z	0	1	0	1	0	1	0	1

Binary Decision Diagrams

[Bryant 86]

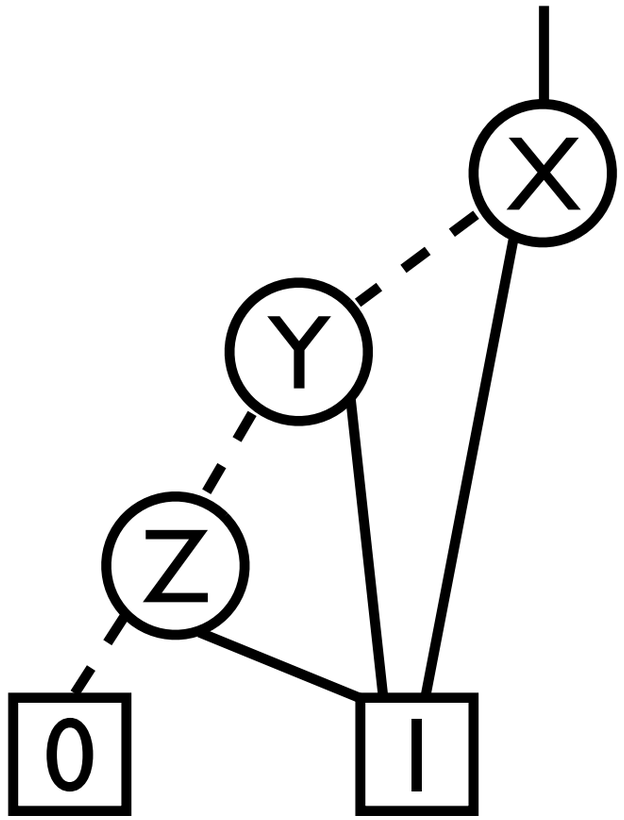


$X \vee Y \vee Z$

X	0	0	0	0	1	1	1	1
Y	0	0	1	1	0	0	1	1
Z	0	1	0	1	0	1	0	1

Binary Decision Diagrams

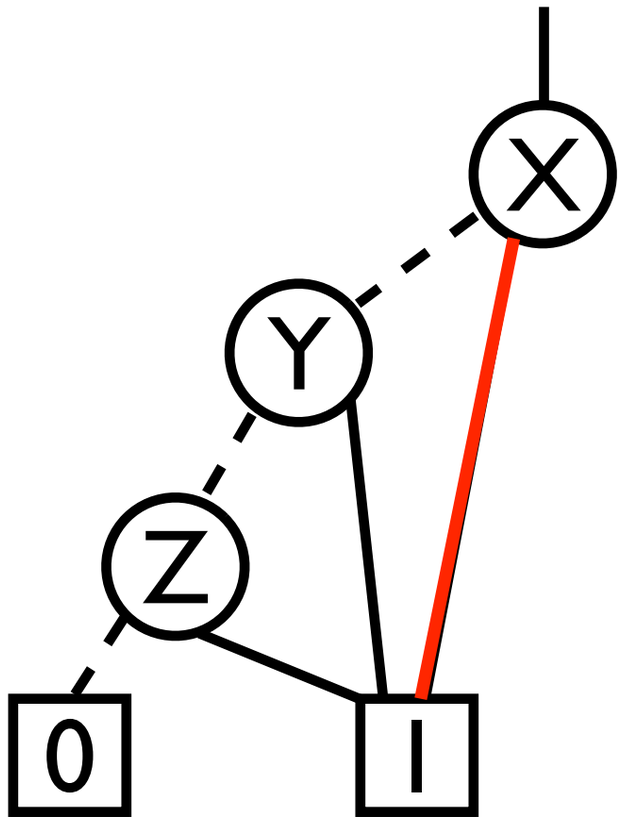
[Bryant 86]



$X \vee Y \vee Z$

X	0	0	0	0	1	1	1	1
Y	0	0	1	1	0	0	1	1
Z	0	1	0	1	0	1	0	1

Binary Decision Diagrams
[Bryant 86]

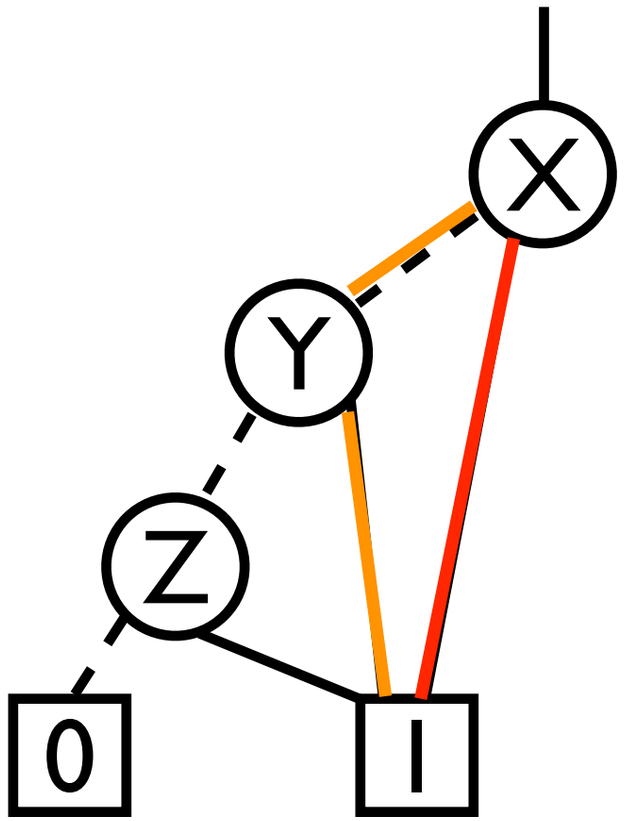


$X \vee Y \vee Z$

$X \vee (\neg X \wedge Y) \vee (\neg X \wedge \neg Y \wedge Z)$

X	0	0	0	0	1	1	1	1
Y	0	0	1	1	0	0	1	1
Z	0	1	0	1	0	1	0	1

Binary Decision Diagrams
[Bryant 86]

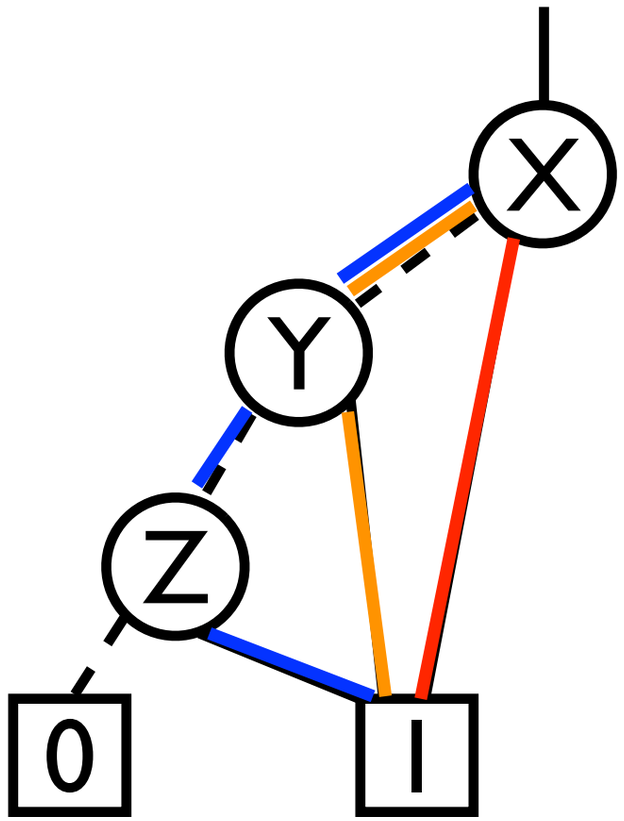


$X \vee Y \vee Z$

$X \vee (\neg X \wedge Y) \vee (\neg X \wedge \neg Y \wedge Z)$

X	0	0	0	0	1	1	1	1
Y	0	0	1	1	0	0	1	1
Z	0	1	0	1	0	1	0	1

Binary Decision Diagrams
[Bryant 86]

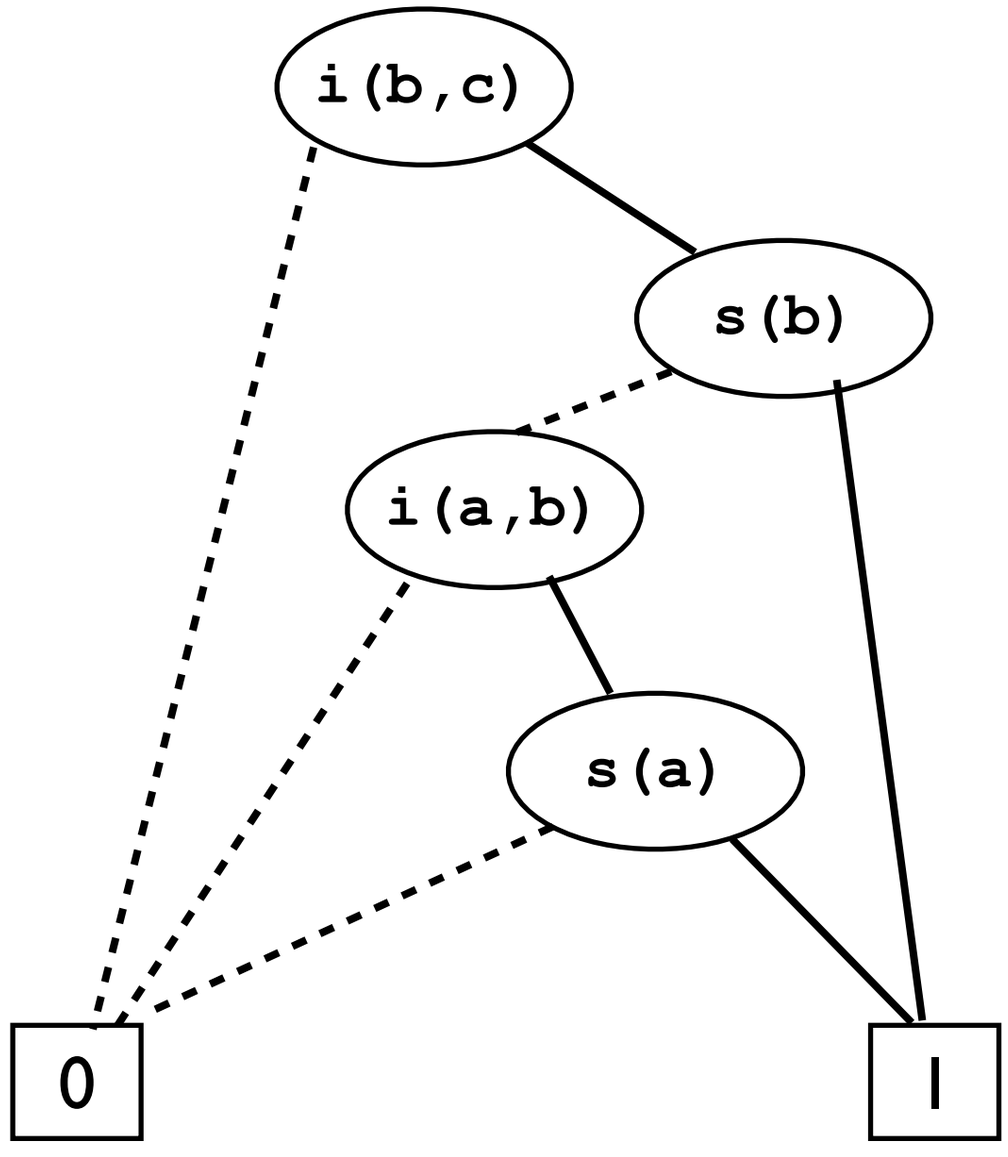


$$X \vee Y \vee Z$$

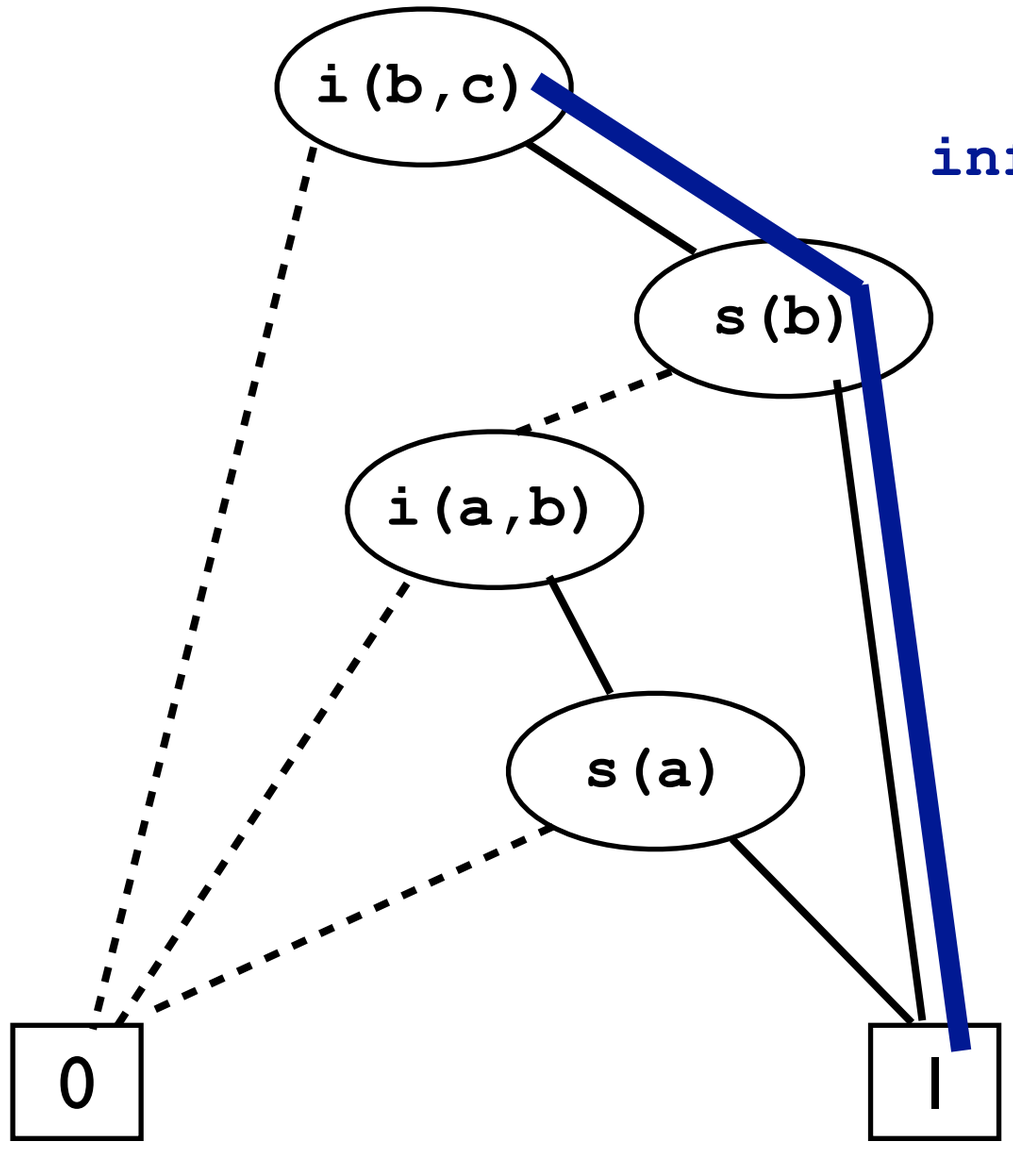
$$X \vee (\neg X \wedge Y) \vee (\neg X \wedge \neg Y \wedge Z)$$

X	0	0	0	0	1	1	1	1
Y	0	0	1	1	0	0	1	1
Z	0	1	0	1	0	1	0	1

Binary Decision Diagrams
[Bryant 86]



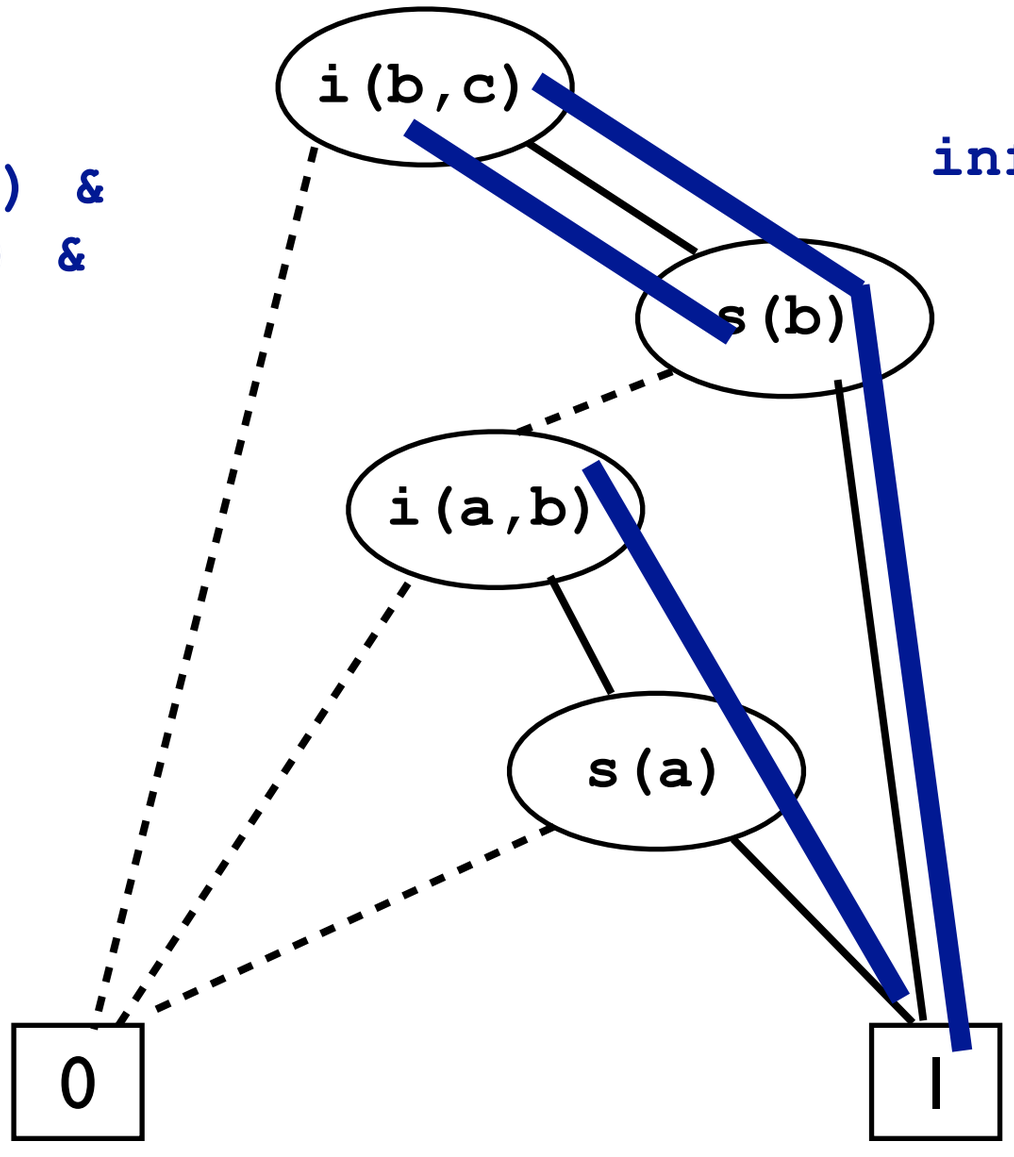
Binary Decision Diagrams
[Bryant 86]



influences(bob,carl) & stress(bob)

Binary Decision Diagrams
[Bryant 86]

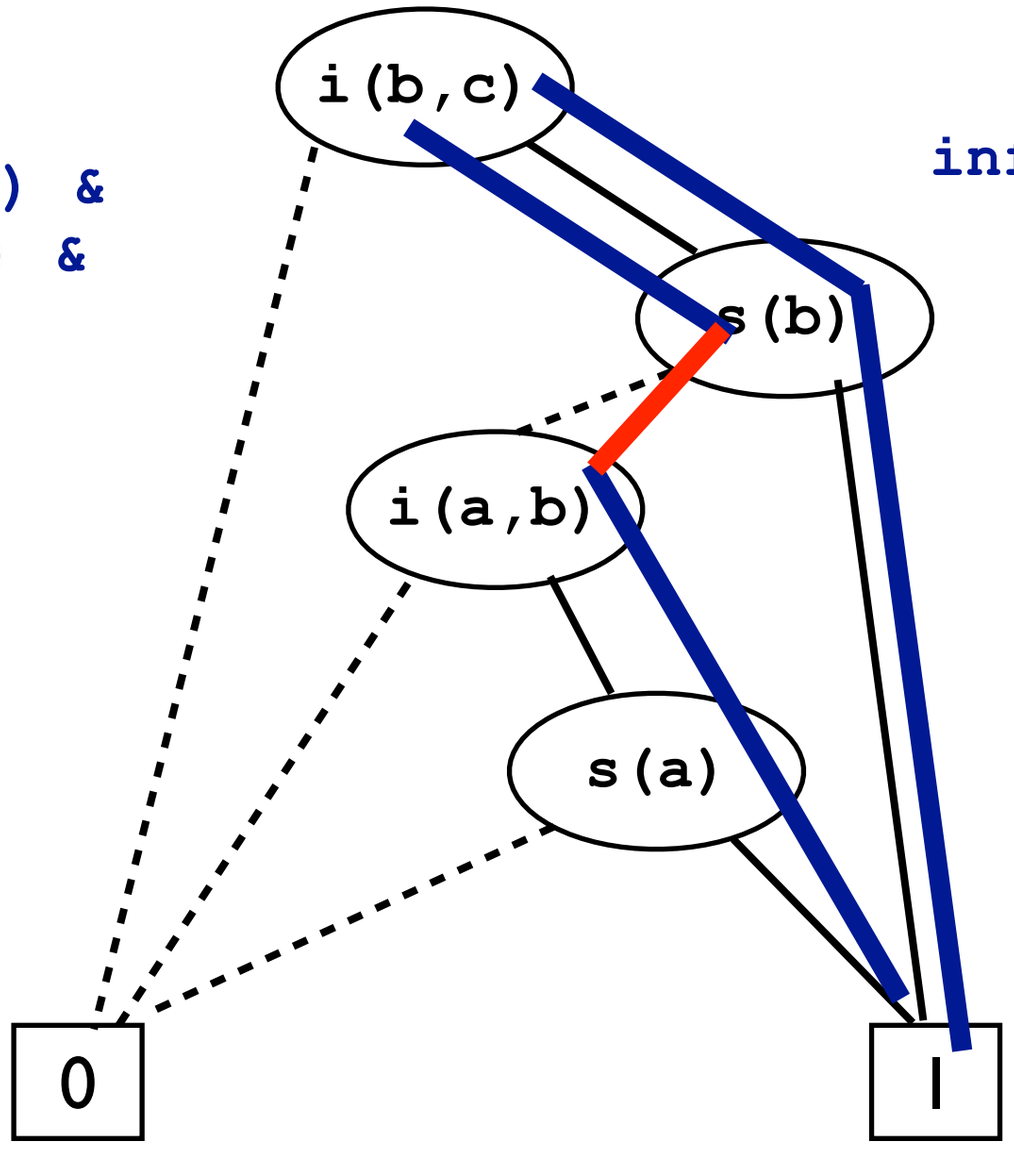
`influences(bob,carl) & influences(ann,bob) & stress(ann)`



`influences(bob,carl) & stress(bob)`

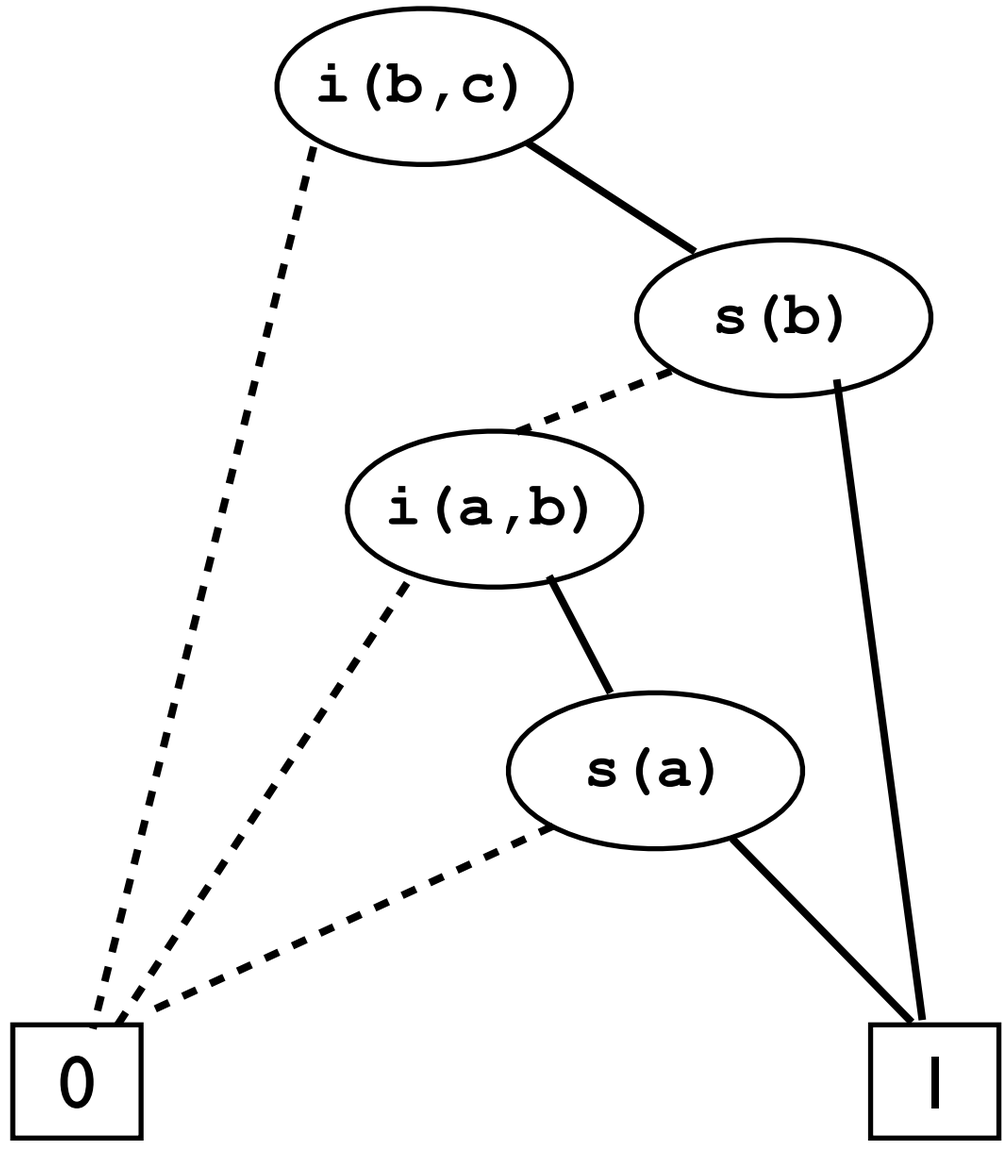
Binary Decision Diagrams
[Bryant 86]

`influences(bob,carl) & influences(ann,bob) & stress(ann) & not stress(bob)`

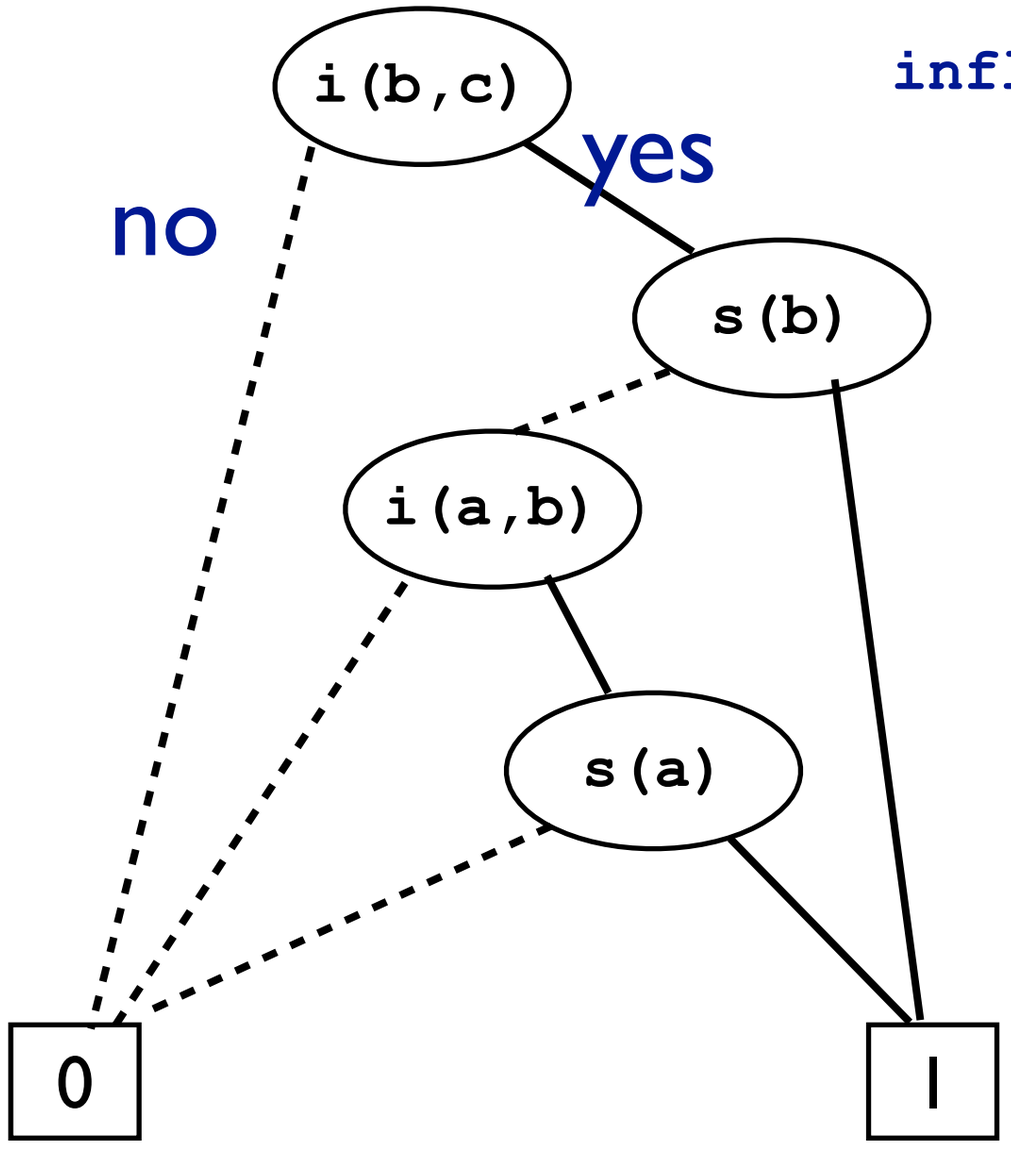


`influences(bob,carl) & stress(bob)`

Binary Decision Diagrams

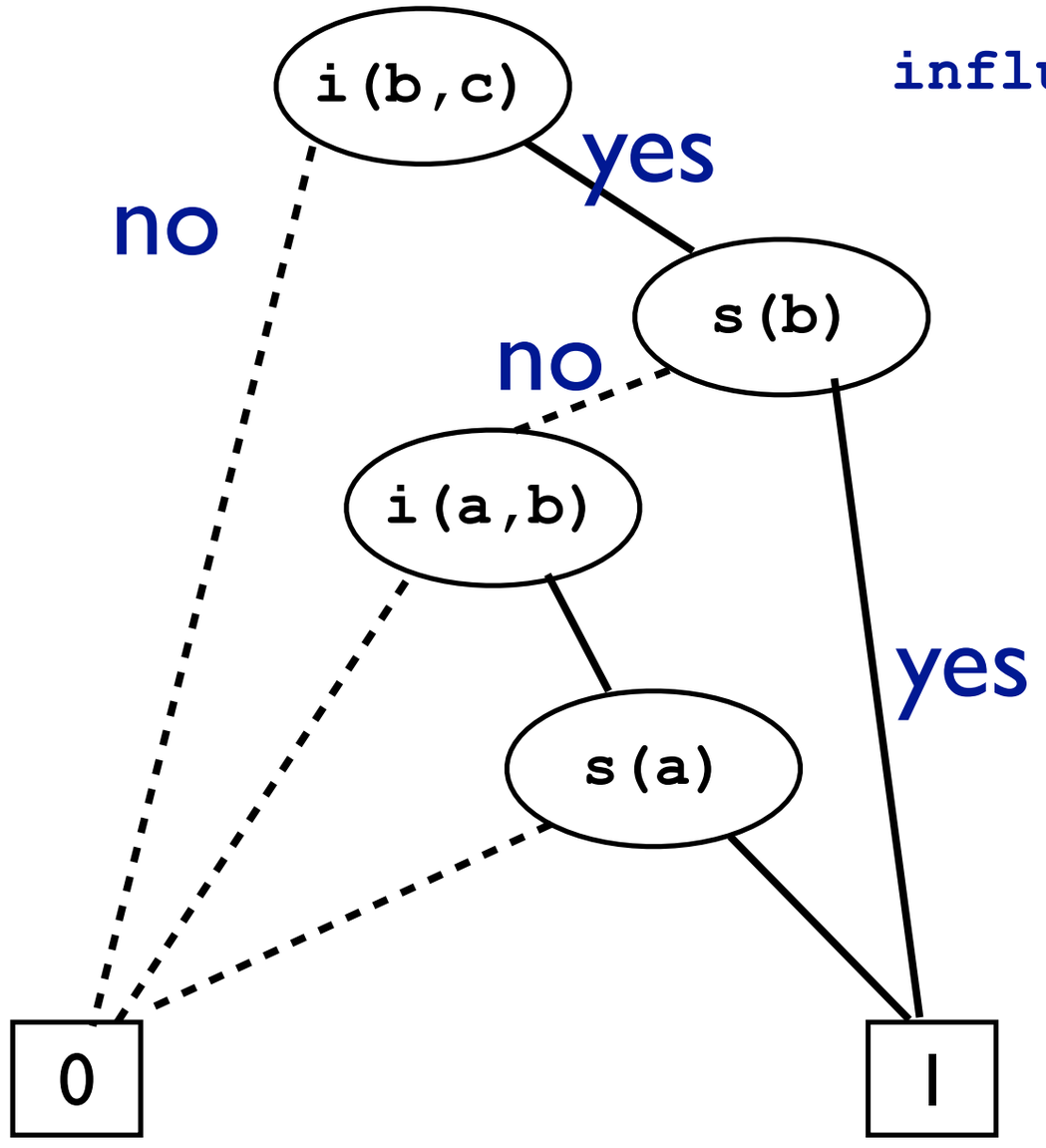


Binary Decision Diagrams



influences(bob,carl)?

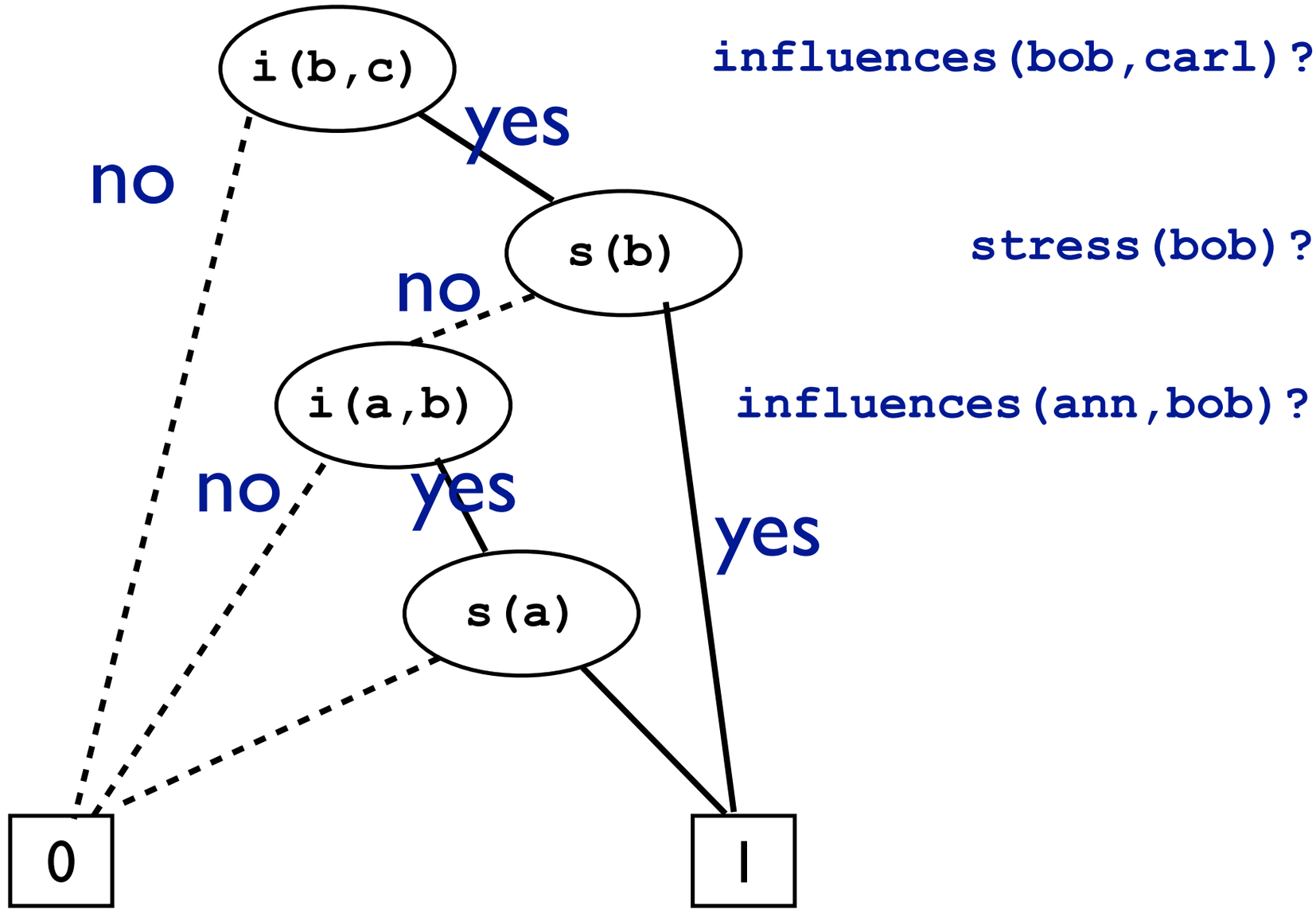
Binary Decision Diagrams



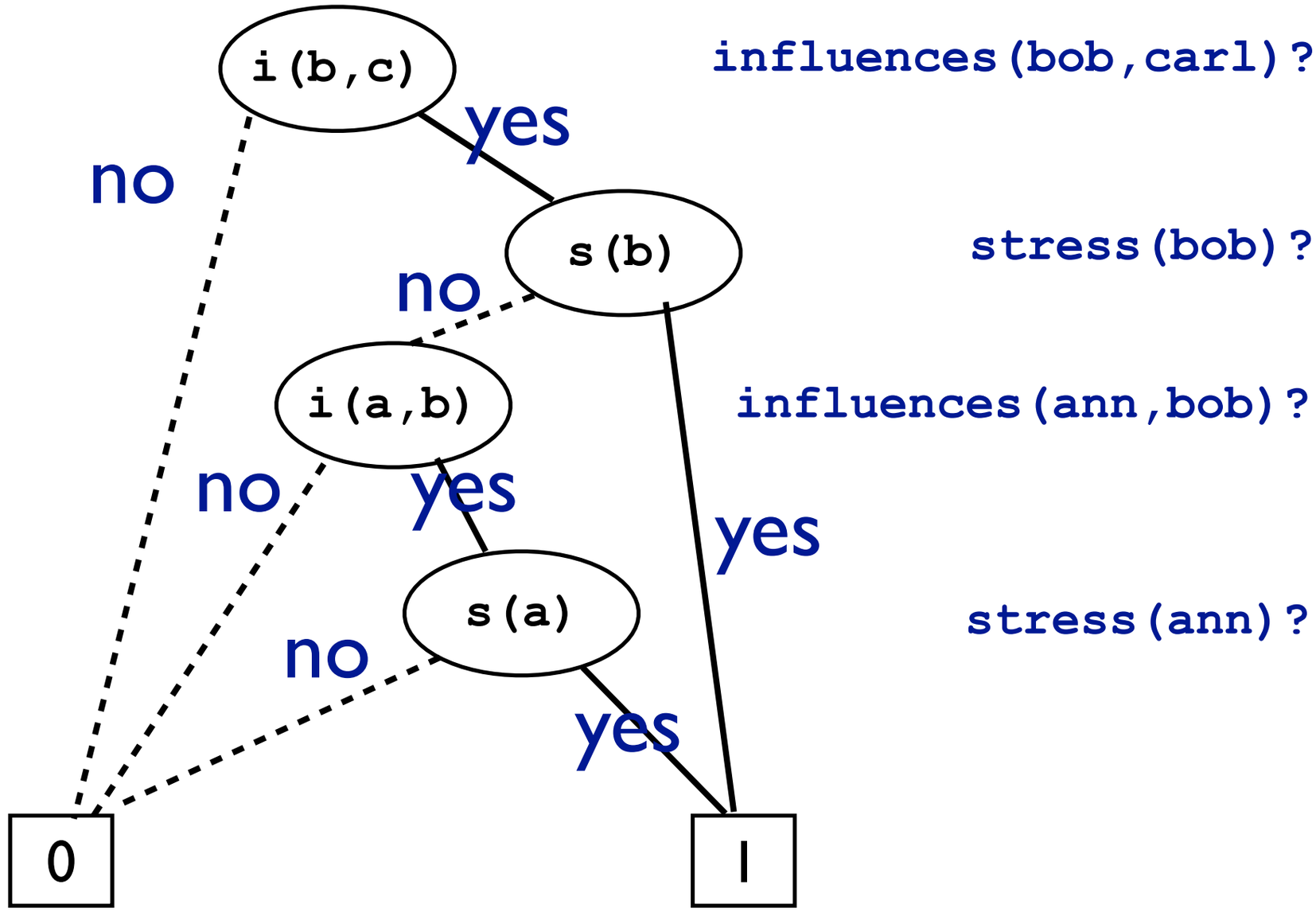
influences(bob,carl)?

stress(bob)?

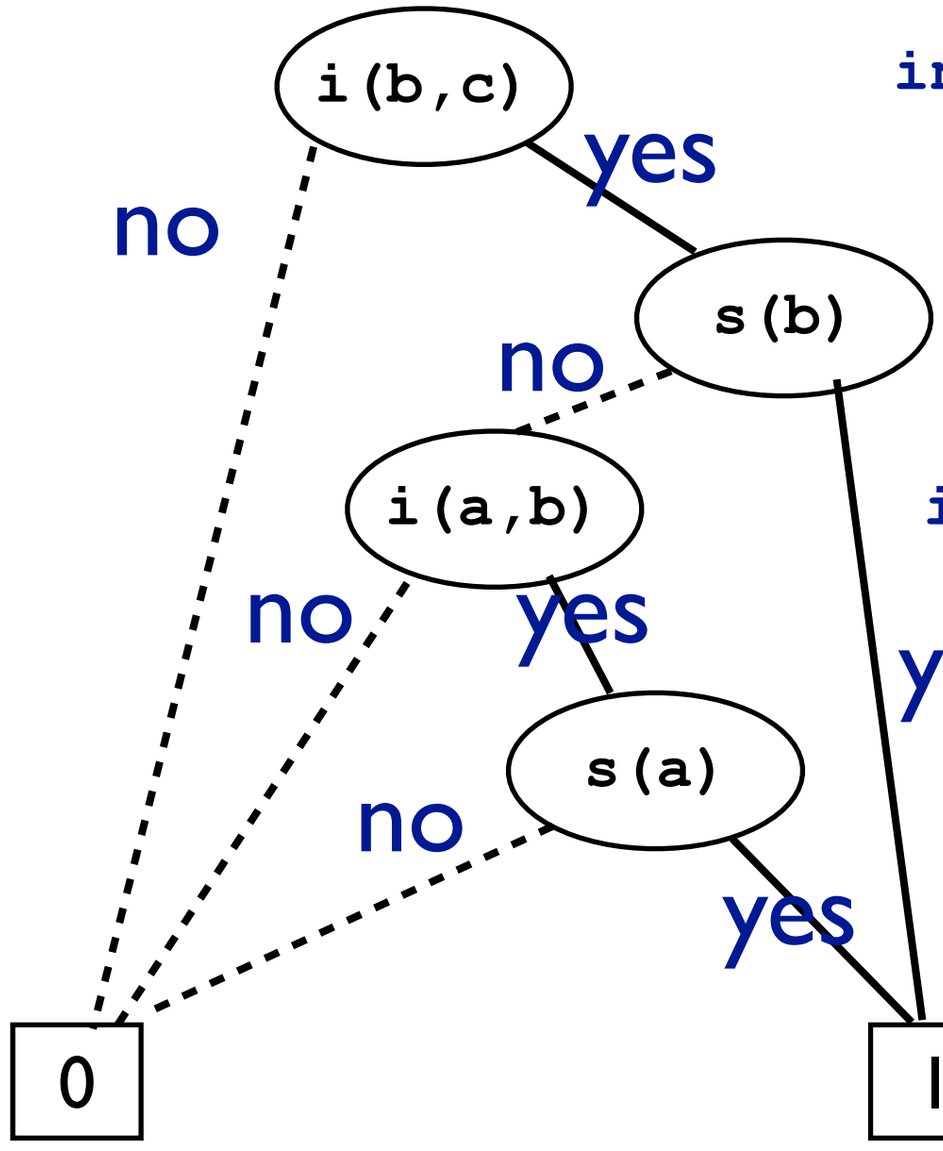
Binary Decision Diagrams



Binary Decision Diagrams



Binary Decision Diagrams



influences(bob,carl)?

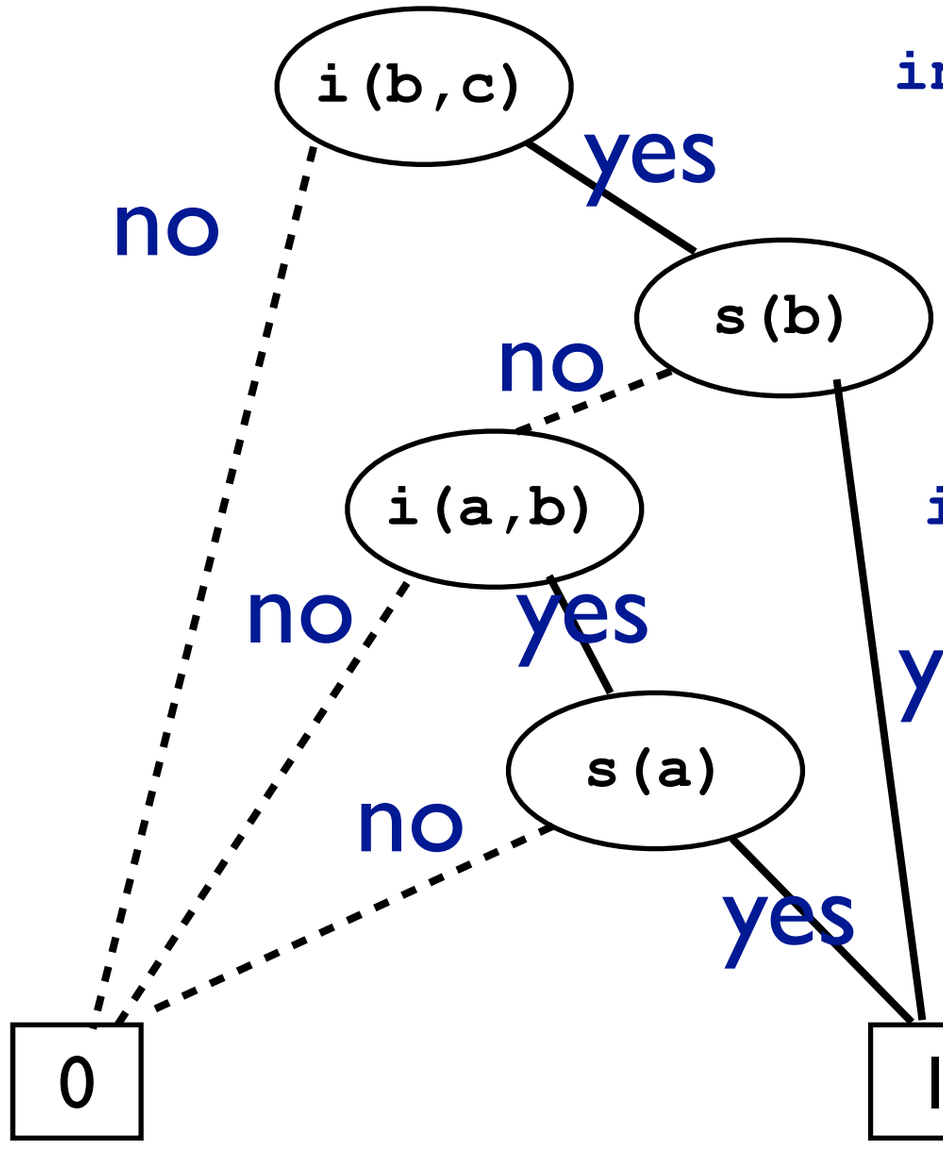
stress(bob)?

influences(ann,bob)?

stress(ann)?

smokes(c) = i(b,c)&s(b) v
i(b,c)&i(a,b)&s(a)

Binary Decision Diagrams



influences(bob,carl)?

stress(bob)?

influences(ann,bob)?

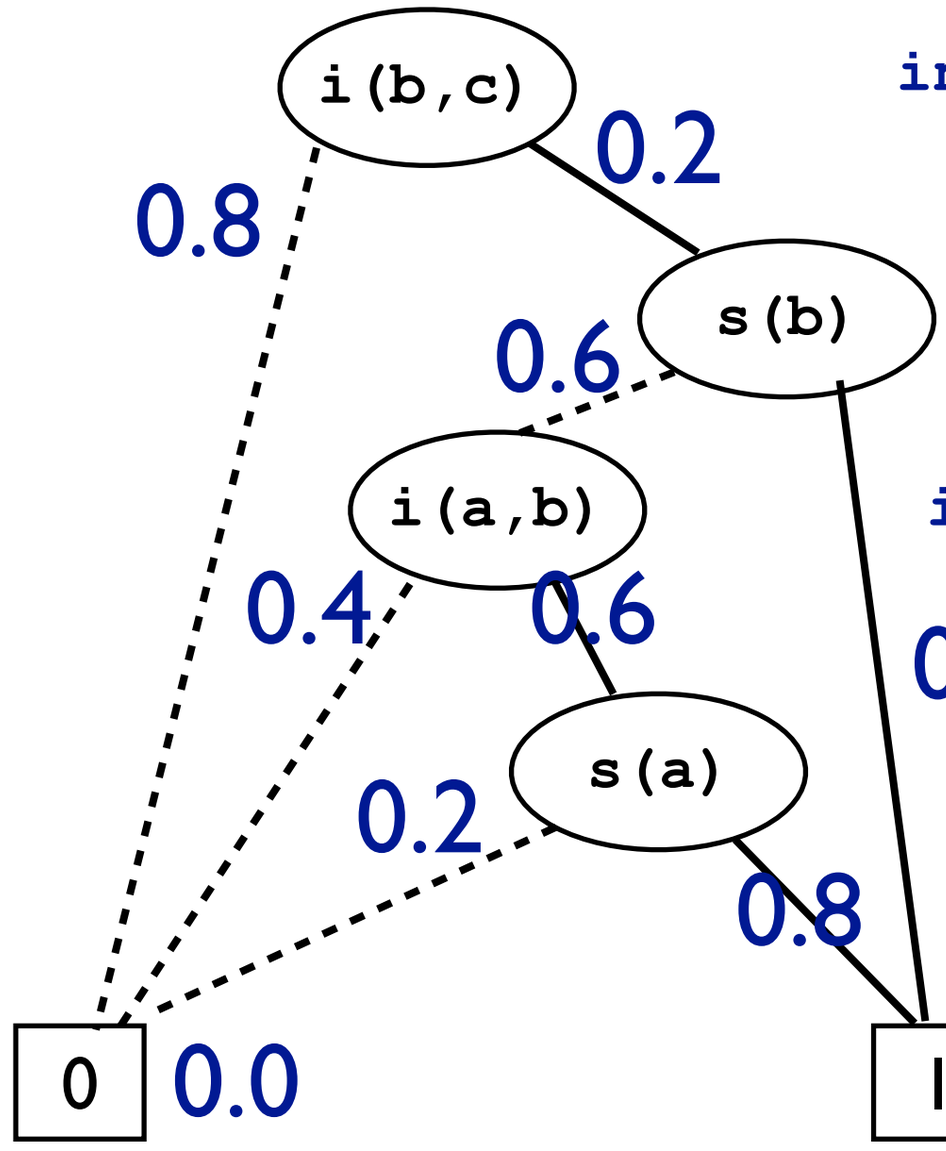
stress(ann)?

probability of

smokes(c)?

smokes(c) = i(b,c)&s(b) v
i(b,c)&i(a,b)&s(a)

Binary Decision Diagrams



`influences(bob,carl)?`

`stress(bob)?`

`influences(ann,bob)?`

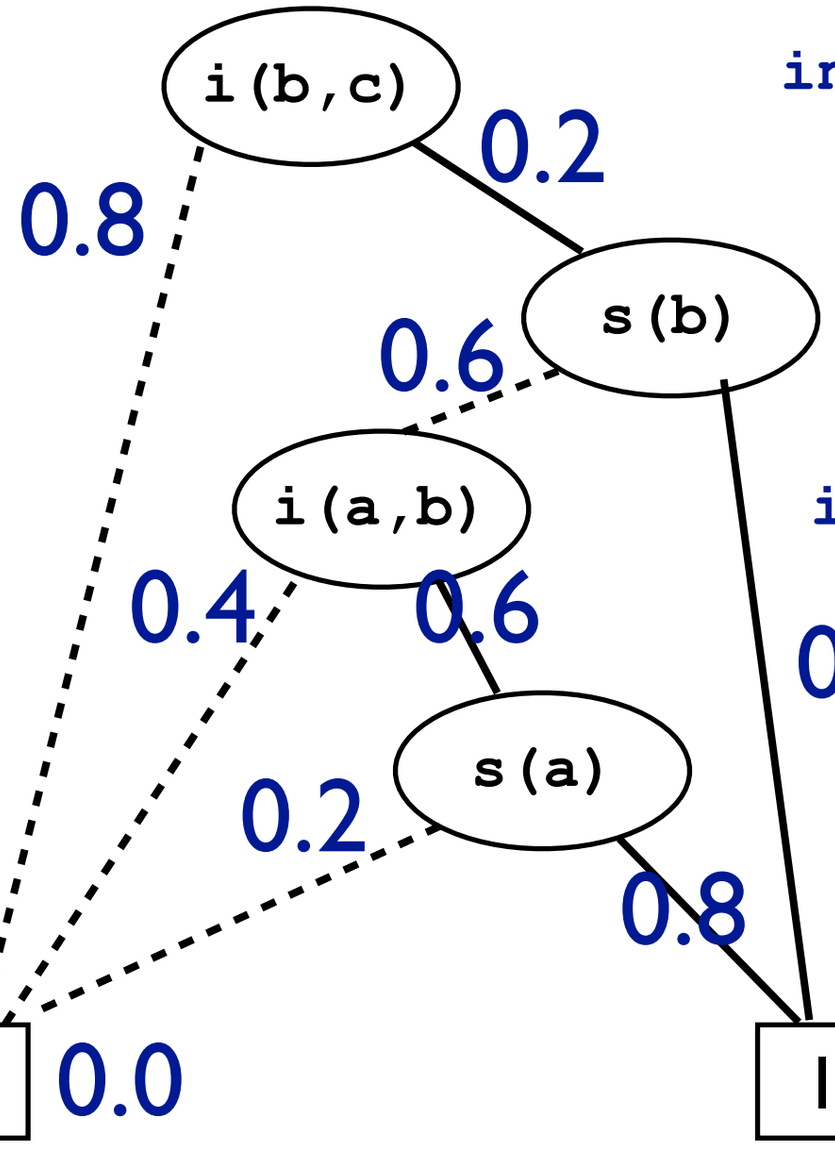
`stress(ann)?`

probability of

`smokes(c)?`

`smokes(c) = i(b,c)&s(b) v`
`i(b,c)&i(a,b)&s(a)`

Binary Decision Diagrams



`influences(bob,carl)?`

`stress(bob)?`

`influences(ann,bob)?`

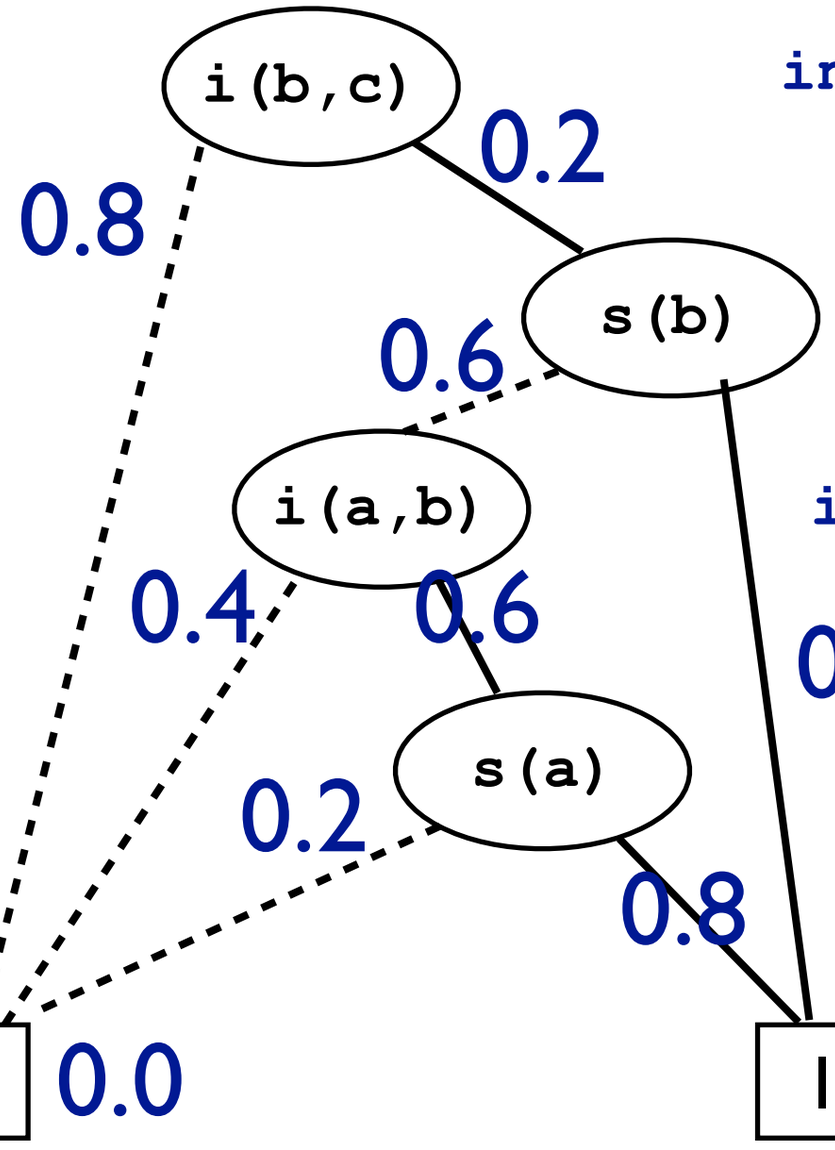
`stress(ann)?`

0.2×0.0+0.8×1.0=0.8

probability of `smokes(c)`?

`smokes(c) = i(b,c)&s(b) v i(b,c)&i(a,b)&s(a)`

Binary Decision Diagrams



`influences(bob,carl)?`

`stress(bob)?`

`influences(ann,bob)?`

`stress(ann)?`

0.4×0.0+0.6×0.8=0.48

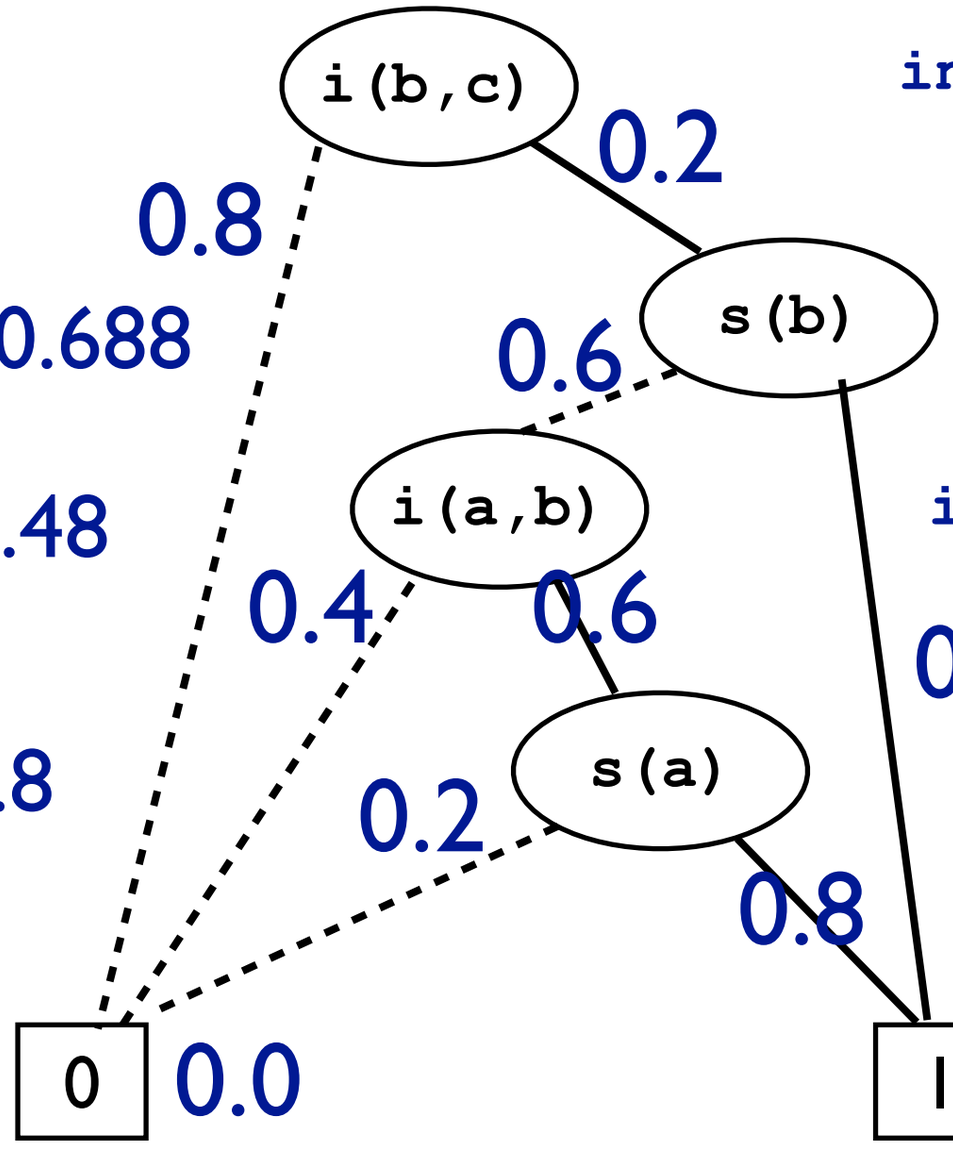
0.2×0.0+0.8×1.0=0.8

probability of

`smokes(c)?`

`smokes(c) = i(b,c)&s(b) v`
`i(b,c)&i(a,b)&s(a)`

Binary Decision Diagrams



influences(bob,carl)?

0.6×0.48+0.4×1.0=0.688

stress(bob)?

0.4×0.0+0.6×0.8=0.48

influences(ann,bob)?

0.2×0.0+0.8×1.0=0.8

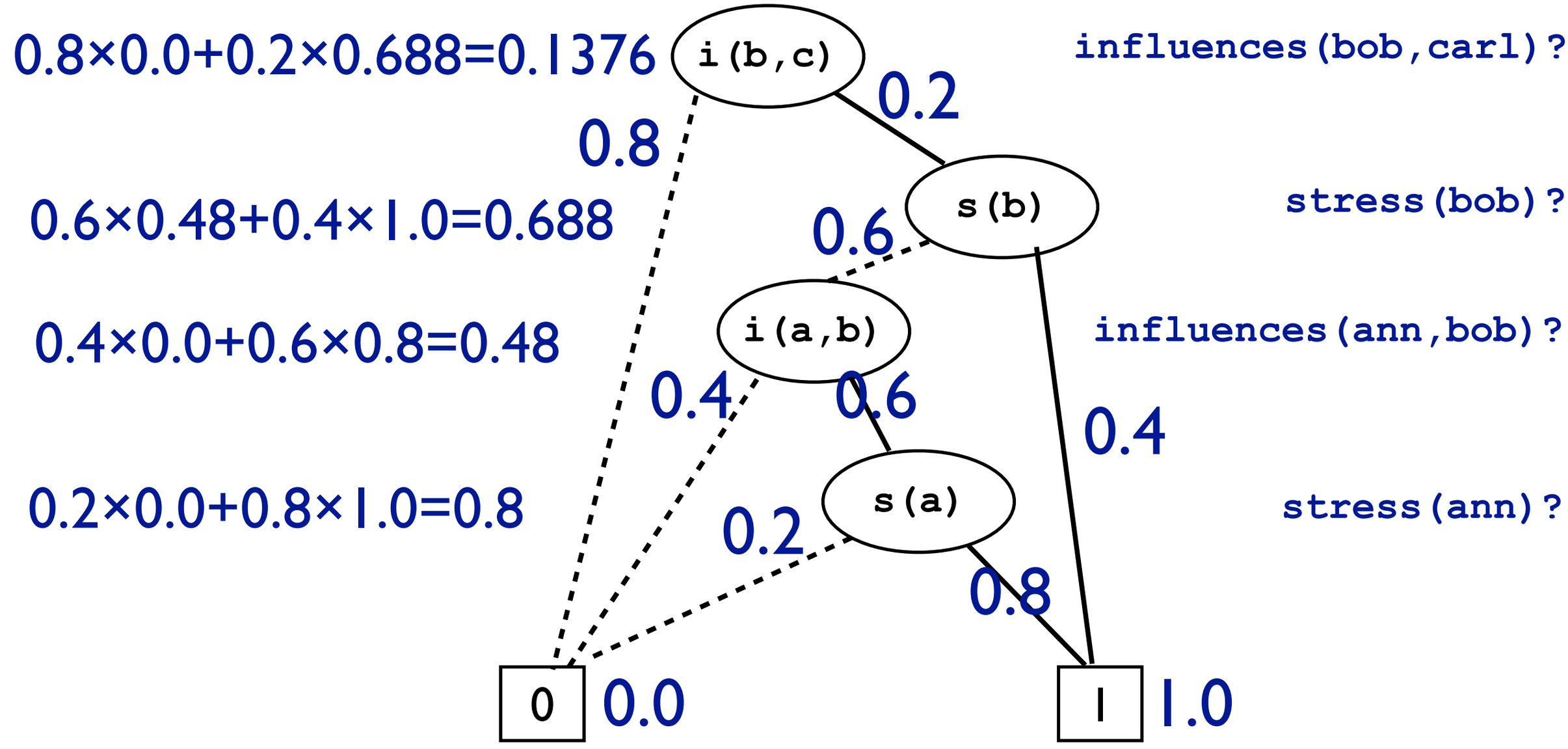
stress(ann)?

probability of

smokes(c)?

smokes(c) = i(b,c)&s(b) v
i(b,c)&i(a,b)&s(a)

Binary Decision Diagrams

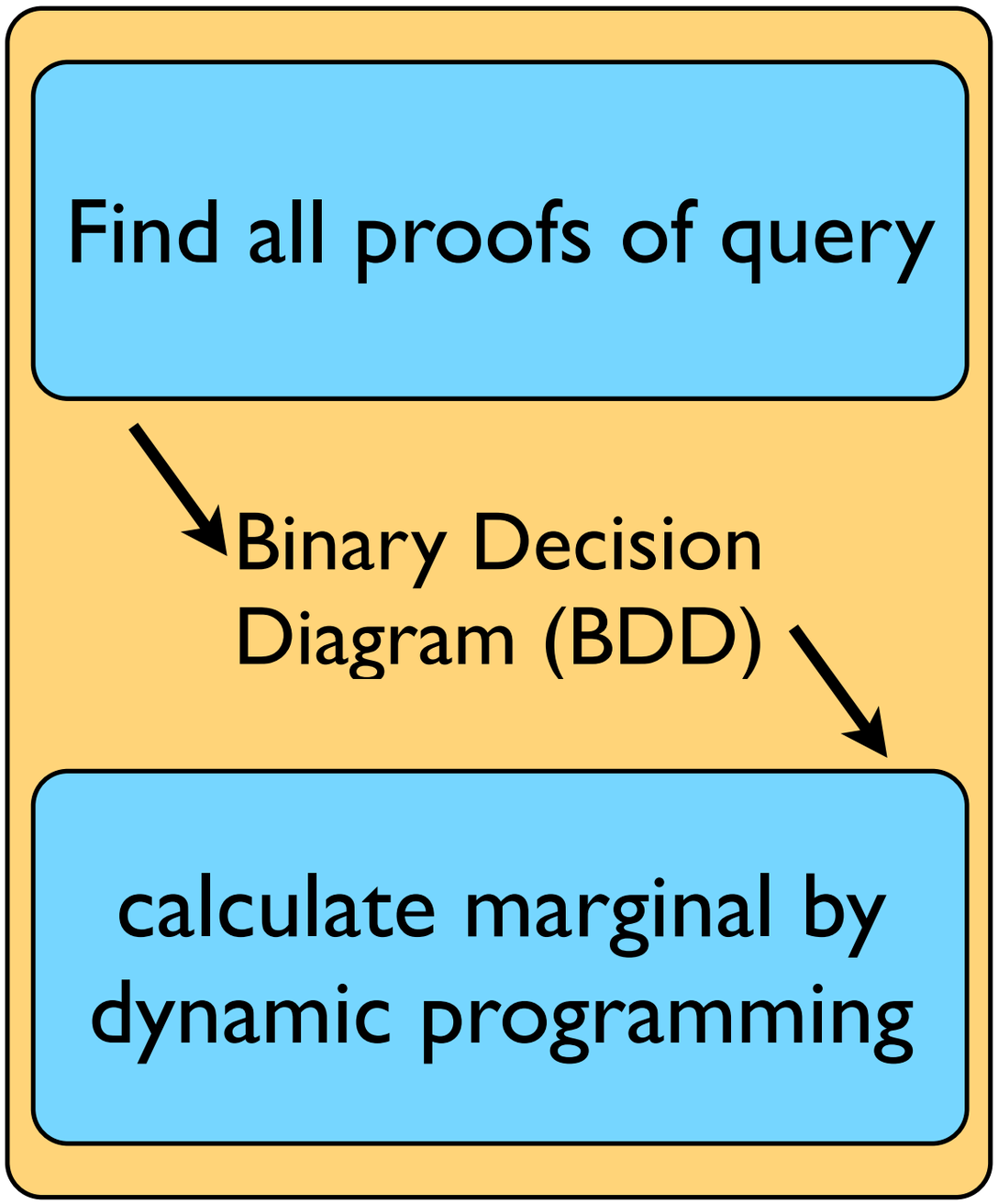


Knowledge Compilation

- Compile once - query many times …
- The knowledge compilation map (Darwiche and Marquis, JAIR 2002)
- Different representations, computational complexity of different operations, … (and, or, condition, ….)
- in StarAI — OBDDs, d-DNNFs, SDD (Darwiche), …
- Also state of the art for Bayesian net inference.

Initial Approach

(ProbLog1 & others)

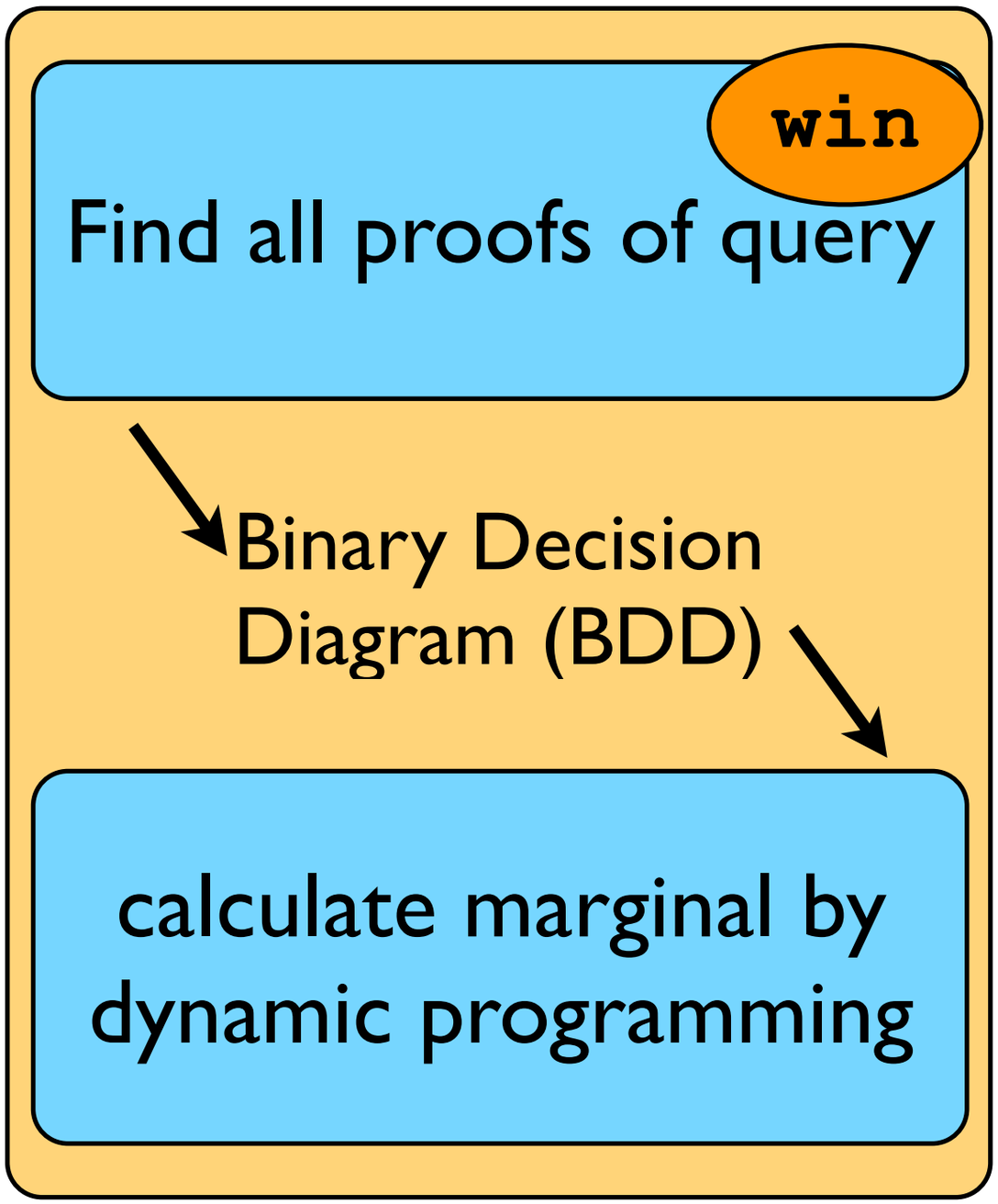


[De Raedt et al, IJCAI 07; Kimmig et al, TPLP 11]

Initial Approach

(ProbLog1 & others)

```
0.4::heads(1).
0.7::heads(2).
0.5::heads(3).
win :- heads(1).
win :- heads(2),heads(3).
```

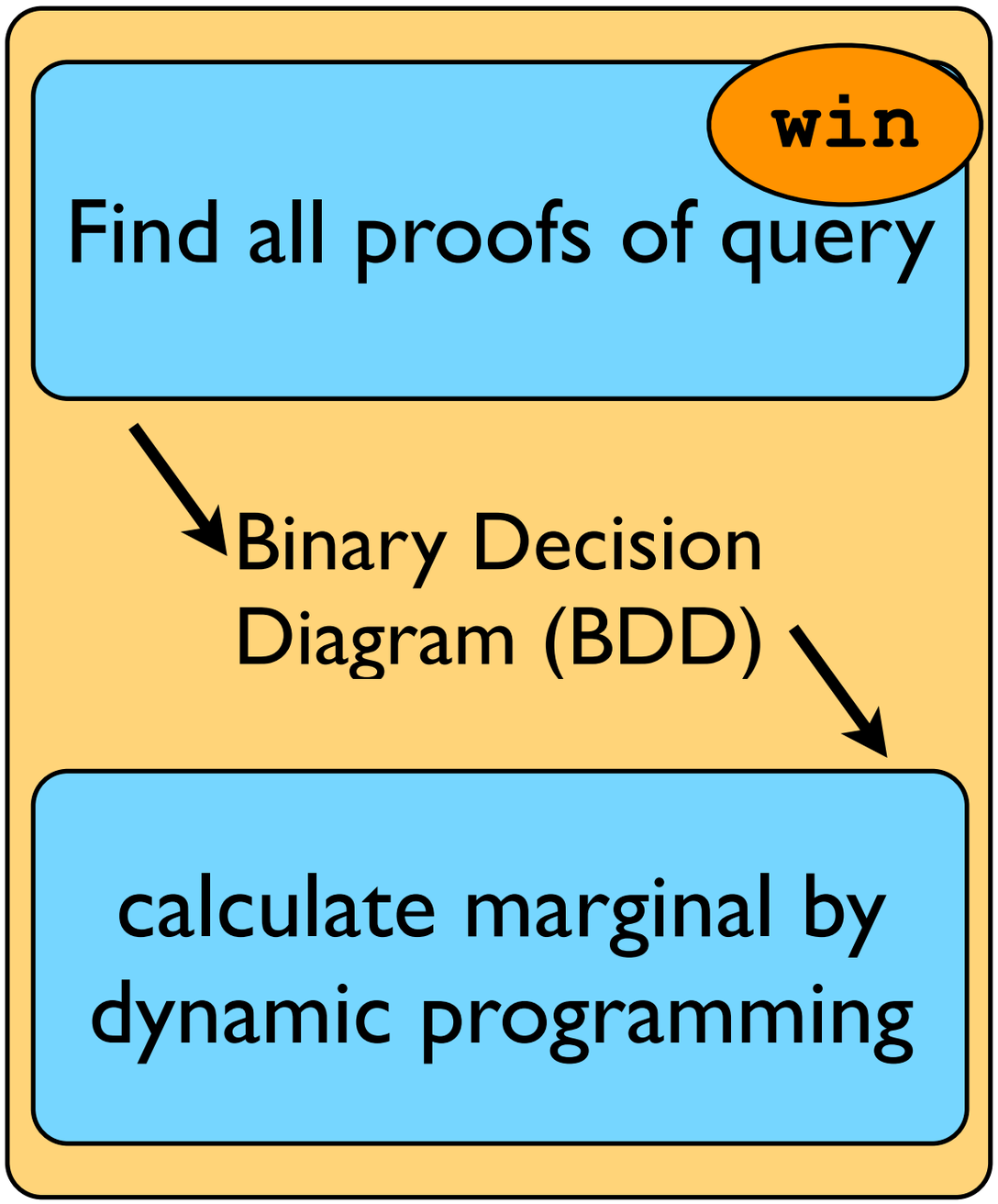


 [De Raedt et al, IJCAI 07; Kimmig et al, TPLP 11]

Initial Approach

(ProbLog1 & others)

```
0.4::heads(1).
0.7::heads(2).
0.5::heads(3).
win :- heads(1).
win :- heads(2),heads(3).
```



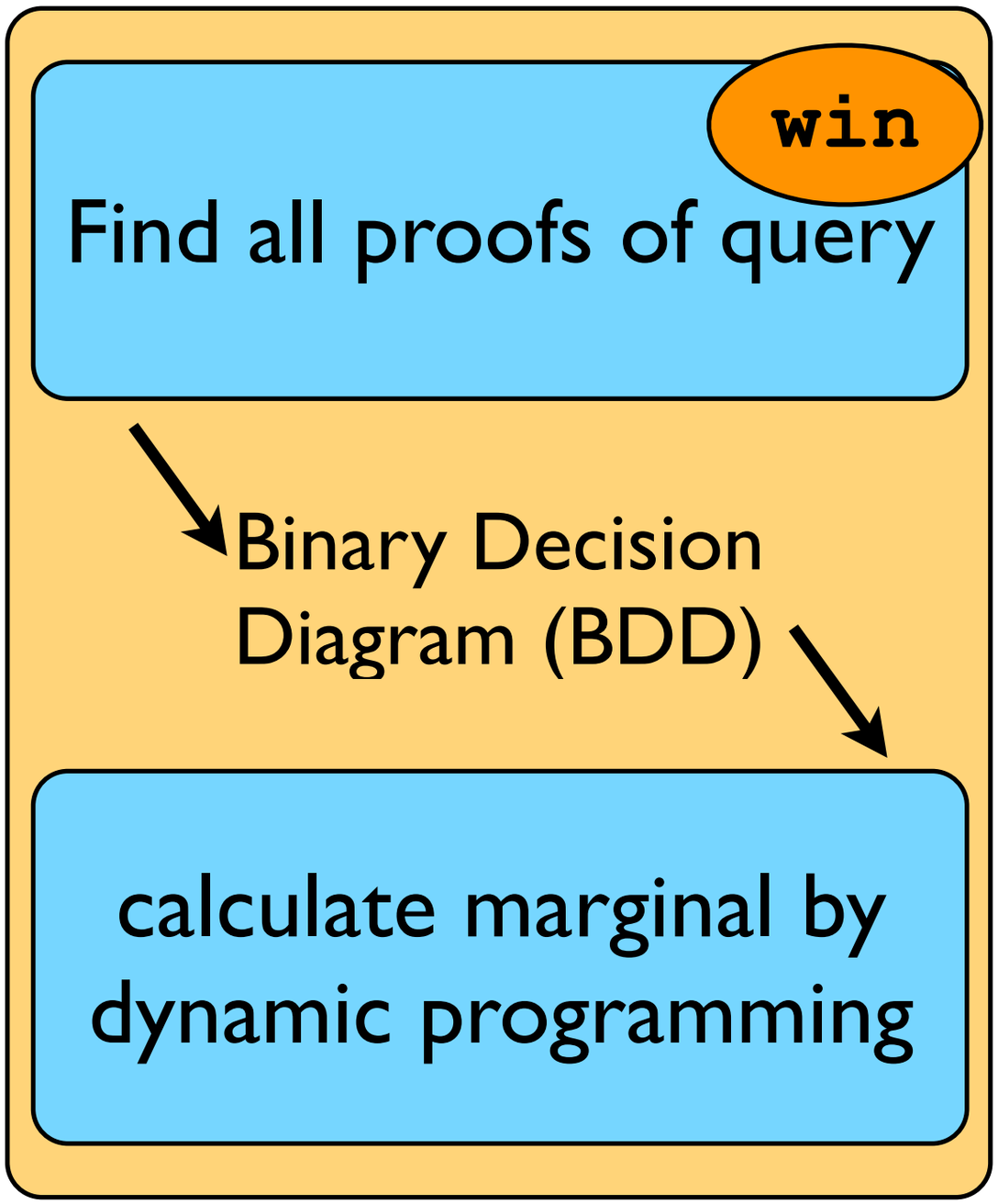

```
heads(1)
heads(2)  & heads(3)
```


[De Raedt et al, IJCAI 07; Kimmig et al, TPLP 11]

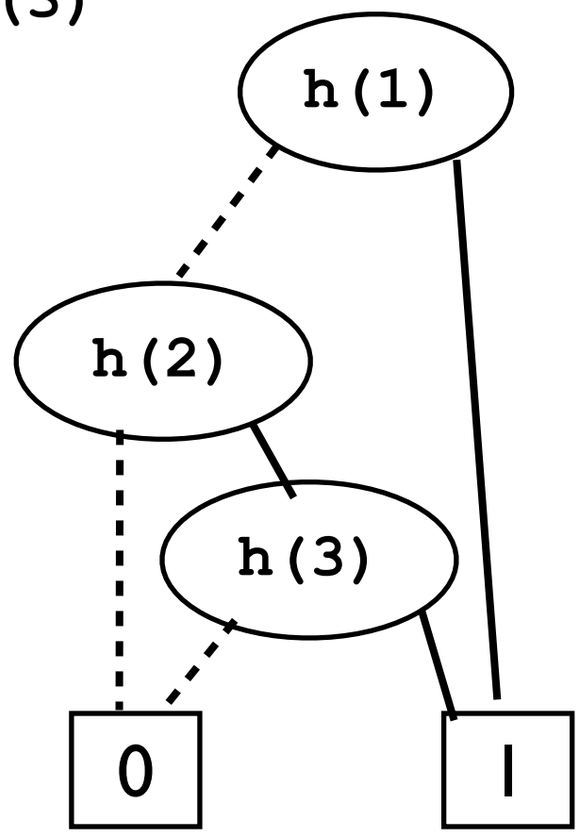
Initial Approach

(ProbLog1 & others)

```
0.4::heads(1).
0.7::heads(2).
0.5::heads(3).
win :- heads(1).
win :- heads(2),heads(3).
```




```
heads(1)
heads(2)  &  heads(3)
```

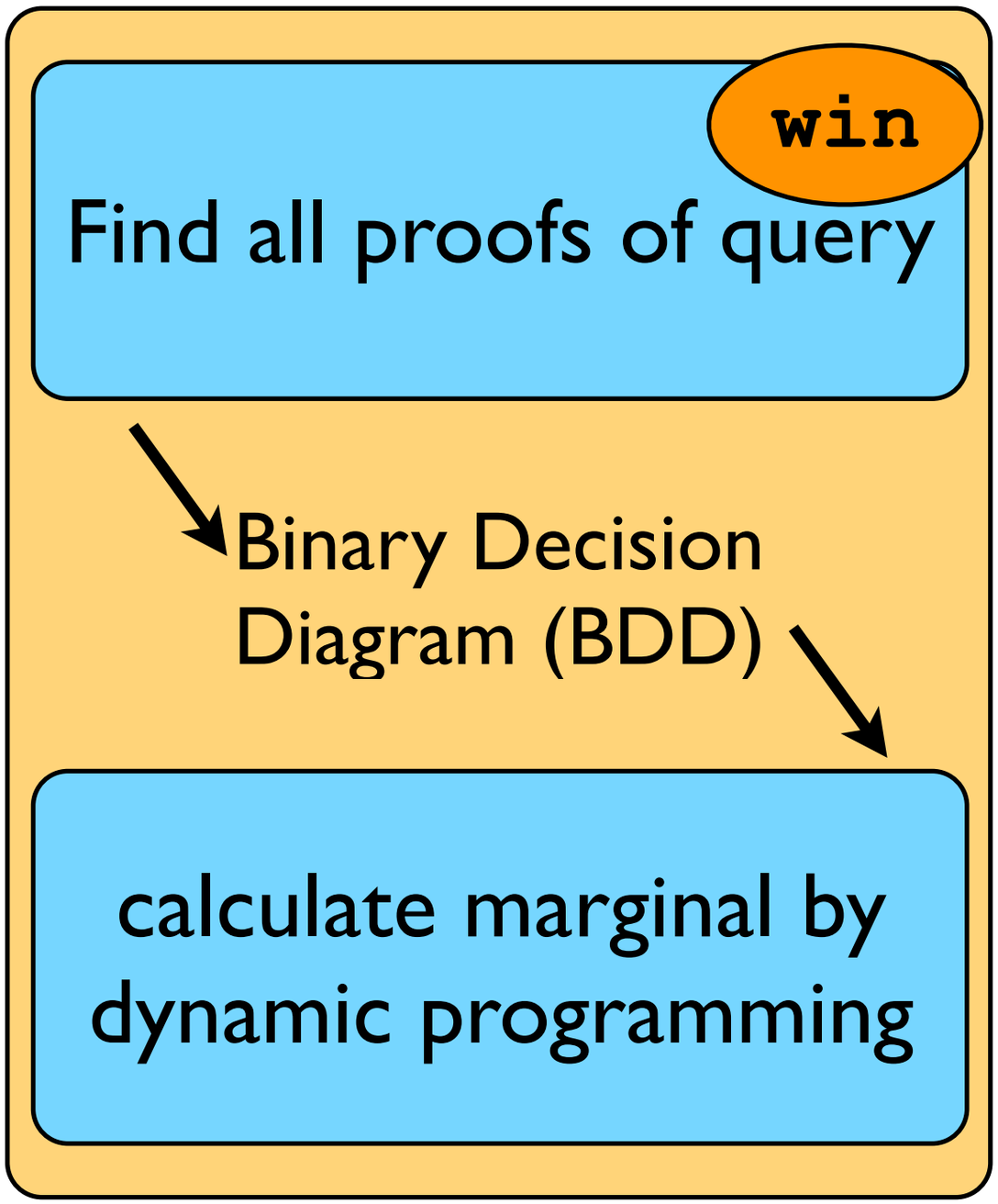


[De Raedt et al, IJCAI 07; Kimmig et al, TPLP 11]

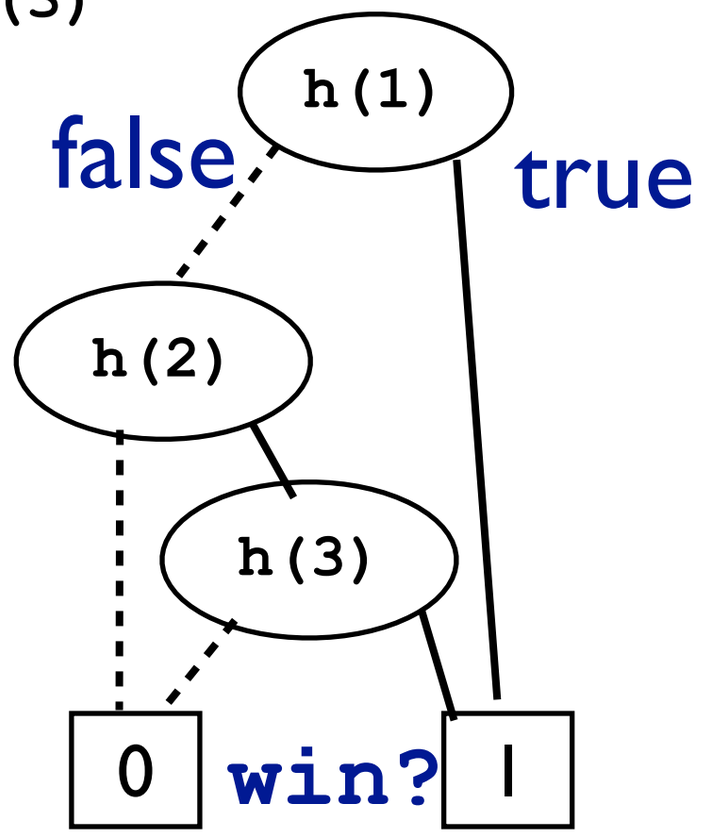
Initial Approach

(ProbLog1 & others)

```
0.4::heads(1).
0.7::heads(2).
0.5::heads(3).
win :- heads(1).
win :- heads(2),heads(3).
```




```
heads(1)
heads(2)  & heads(3)
```

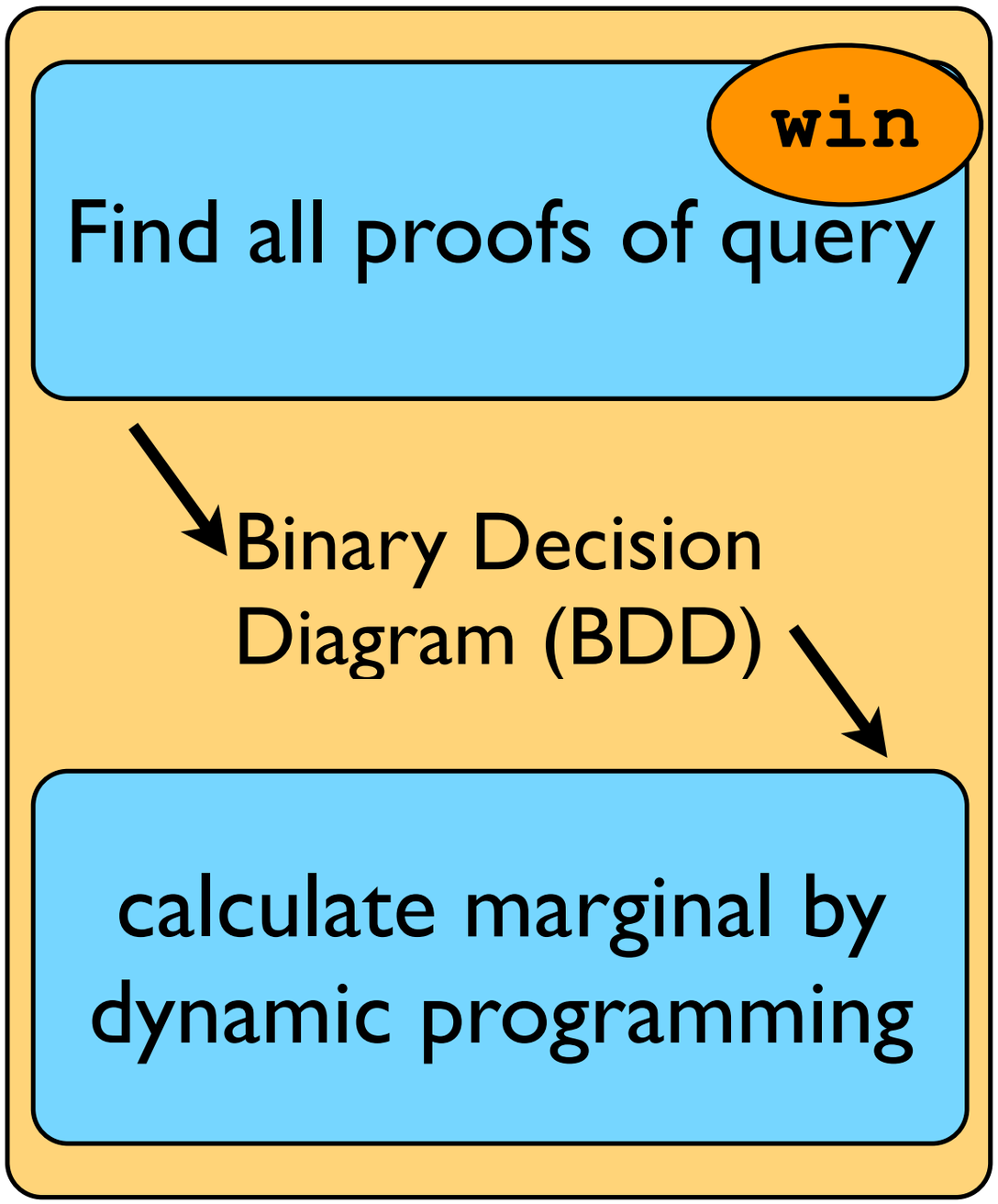


[De Raedt et al, IJCAI 07; Kimmig et al, TPLP 11]

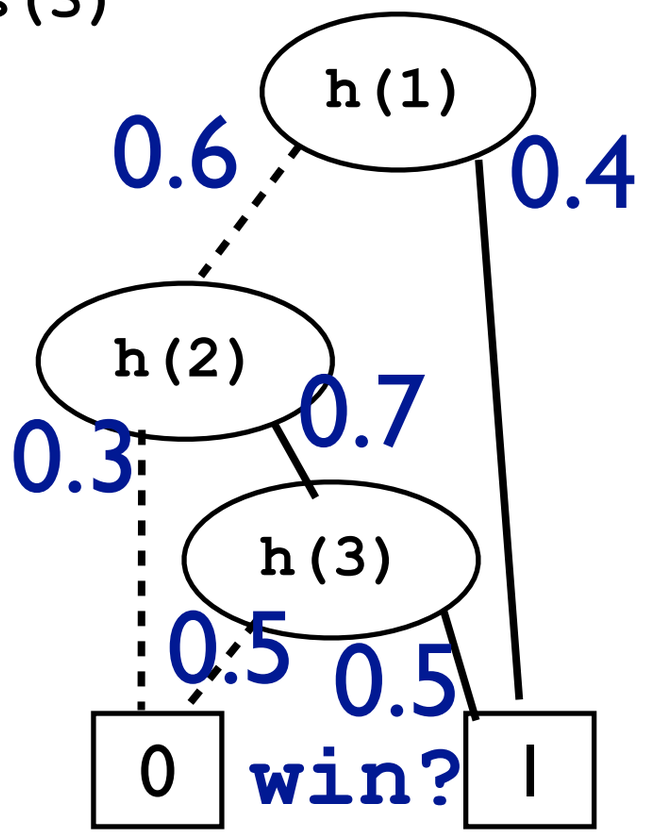
Initial Approach

(ProbLog1 & others)

```
0.4::heads(1).
0.7::heads(2).
0.5::heads(3).
win :- heads(1).
win :- heads(2),heads(3).
```




```
heads(1)
heads(2)  & heads(3)
```



P(**win**) = probability of reaching 1-leaf

 [De Raedt et al, IJCAI 07; Kimmig et al, TPLP 11]

Answering Questions

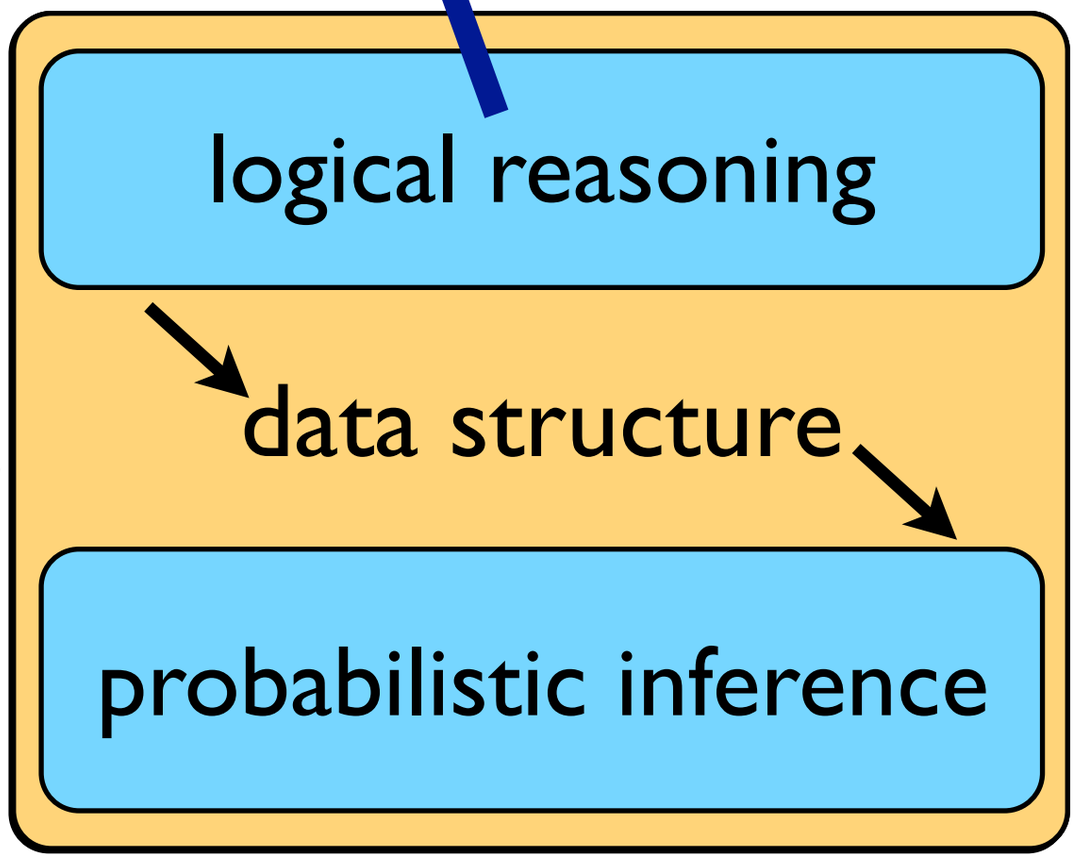
1. using proofs
2. using models

Given:

program

queries

evidence



Find:

marginal probabilities

conditional probabilities

MPE state

Logical Reasoning: Find a model (SAT) Here : prolog

```
stress(ann).
influences(ann,bob).
influences(bob,carl).

smokes(X) :- stress(X).
smokes(X) :-
      influences(Y,X),
      smokes(Y).
```

```
?- smokes(carl).
```

- Forward reasoning to construct unique model:

Logical Reasoning: Find a model (SAT) Here : prolog

```
stress(ann).
influences(ann,bob).
influences(bob,carl).

smokes(X) :- stress(X).
smokes(X) :-
      influences(Y,X),
      smokes(Y).
```

```
?- smokes(carl).
```

- Forward reasoning to construct unique model:
  - Start with database facts

```
stress(ann).
influences(ann,bob).
influences(bob,carl).
```

Logical Reasoning: Find a model (SAT) Here : prolog

```
stress(ann).
influences(ann,bob).
influences(bob,carl).

smokes(X) :- stress(X).
smokes(X) :-
      influences(Y,X),
      smokes(Y).
```

```
?- smokes(carl).
```

- Forward reasoning to construct unique model:
  - Start with database facts
  - Use rules to add more facts

```
stress(ann).
influences(ann,bob).
influences(bob,carl).

smokes(ann).
```

Logical Reasoning: Find a model (SAT) Here : prolog

```
stress(ann).
influences(ann,bob).
influences(bob,carl).

smokes(X) :- stress(X).
smokes(X) :-
      influences(Y,X),
      smokes(Y).
```

```
?- smokes(carl).
```

- Forward reasoning to construct unique model:
  - Start with database facts

```
stress(ann).
influences(ann,bob).
influences(bob,carl).
```

  - Use rules to add more facts

```
smokes(ann).
smokes(bob).
```

Logical Reasoning: Find a model (SAT) Here : prolog

```
stress(ann).
influences(ann,bob).
influences(bob,carl).

smokes(X) :- stress(X).
smokes(X) :-
      influences(Y,X),
      smokes(Y).
```

```
?- smokes(carl).
```

- Forward reasoning to construct unique model:
  - Start with database facts

```
stress(ann).
influences(ann,bob).
influences(bob,carl).
```

  - Use rules to add more facts

```
smokes(ann).
smokes(bob).
smokes(carl).
```

Logical Reasoning: Find a model (SAT) Here : prolog

```
stress(ann).
influences(ann,bob).
influences(bob,carl).

smokes(X) :- stress(X).
smokes(X) :-
      influences(Y,X),
      smokes(Y).
```

```
?- smokes(carl).
```

- Forward reasoning to construct unique model:
  - Start with database facts
  - Use rules to add more facts
- Query true iff in model

```
stress(ann).
influences(ann,bob).
influences(bob,carl).

smokes(ann).
smokes(bob).
smokes(carl).
```

Logical Reasoning: Find a model (SAT) Here : prolog

```
stress(ann).
influences(ann,bob).
influences(bob,carl).

smokes(X) :- stress(X).
smokes(X) :-
      influences(Y,X),
      smokes(Y).
```

```
?- smokes(carl).
```

- Forward reasoning to construct unique model:
  - Start with database facts
  - Use rules to add more facts
- Query true iff in model
- **ProbLog**: each possible world is a model, probability of query is sum over models where query is true

```
stress(ann).
influences(ann,bob).
influences(bob,carl).

smokes(ann).
smokes(bob).
smokes(carl).
```

Logical Reasoning: Find a model (SAT) Here : prolog

```
stress(ann).
influences(ann,bob).
influences(bob,carl).

smokes(X) :- stress(X).
smokes(X) :-
      influences(Y,X),
      smokes(Y).
```

```
?- smokes(carl).
```

- Forward reasoning to construct unique model:
  - Start with database facts
  - Use rules to add more facts
- Query true iff in model

```
stress(ann).
influences(ann,bob).
influences(bob,carl).

smokes(ann).
smokes(bob).
smokes(carl).
```

- **ProbLog**: each possible world is a model, probability of query is sum over models where query is true

→ weighted model counting

ProbLog → CNF

```
?- smokes(carl).
```

```
0.8::stress(ann).
0.4::stress(bob).
0.6::influences(ann,bob).
0.2::influences(bob,carl).

smokes(X) :- stress(X).
smokes(X) :-
      influences(Y,X),
      smokes(Y).
```

ProbLog →CNF

```
0.8::stress(ann).
0.4::stress(bob).
0.6::influences(ann,bob).
0.2::influences(bob,carl).

smokes(X) :- stress(X).
smokes(X) :-
      influences(Y,X),
      smokes(Y).
```

```
?- smokes(carl).
```

- Find relevant ground rules by backward reasoning

ProbLog →CNF

```
0.8::stress(ann).
0.4::stress(bob).
0.6::influences(ann,bob).
0.2::influences(bob,carl).

smokes(X) :- stress(X).
smokes(X) :-
     influences(Y,X),
     smokes(Y).
```

```
?- smokes(carl).
```

- Find relevant ground rules by backward reasoning

```
smokes(carl) :- influences(bob,carl),smokes(bob).
smokes(bob) :- stress(bob).
smokes(bob) :- influences(ann,bob),smokes(ann).
smokes(ann) :- stress(ann).
```

ProbLog → CNF

```
0.8::stress(ann).
0.4::stress(bob).
0.6::influences(ann,bob).
0.2::influences(bob,carl).

smokes(X) :- stress(X).
smokes(X) :-
     influences(Y,X),
     smokes(Y).
```

```
?- smokes(carl).
```

- Find relevant ground rules by backward reasoning

```
smokes(carl) :- influences(bob,carl),smokes(bob).
smokes(bob) :- stress(bob).
smokes(bob) :- influences(ann,bob),smokes(ann).
smokes(ann) :- stress(ann).
```

- Convert to propositional logic formula

ProbLog → CNF

```
0.8::stress(ann).
0.4::stress(bob).
0.6::influences(ann,bob).
0.2::influences(bob,carl).

smokes(X) :- stress(X).
smokes(X) :-
     influences(Y,X),
     smokes(Y).
```

```
?- smokes(carl).
```

- Find relevant ground rules by backward reasoning

```
smokes(carl) :- influences(bob,carl),smokes(bob).
smokes(bob)  :- stress(bob).
smokes(bob)  :- influences(ann,bob),smokes(ann).
smokes(ann)  :- stress(ann).
```

- Convert to propositional logic formula

may require loop-breaking

$$sm(c) \leftrightarrow (i(b,c) \wedge sm(b)) \wedge sm(b) \leftrightarrow (st(b) \vee (i(a,b) \wedge sm(a))) \wedge sm(a) \leftrightarrow st(a)$$

ProbLog → CNF

```
0.8::stress(ann).
0.4::stress(bob).
0.6::influences(ann,bob).
0.2::influences(bob,carl).

smokes(X) :- stress(X).
smokes(X) :-
     influences(Y,X),
     smokes(Y).
```

```
?- smokes(carl).
```

- Find relevant ground rules by backward reasoning

```
smokes(carl) :- influences(bob,carl),smokes(bob).
smokes(bob)  :- stress(bob).
smokes(bob)  :- influences(ann,bob),smokes(ann).
smokes(ann)  :- stress(ann).
```

- Convert to propositional logic formula

may require loop-breaking

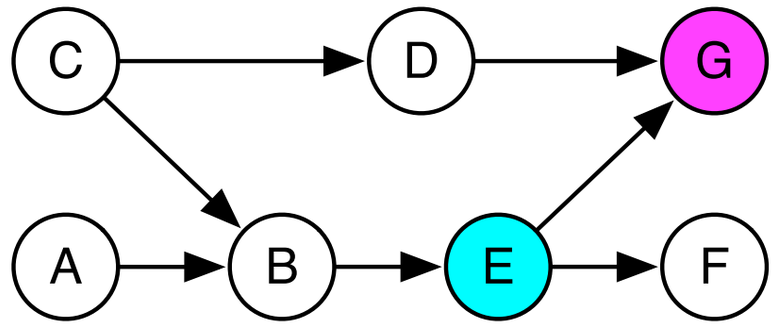
$$sm(c) \leftrightarrow (i(b,c) \wedge sm(b)) \wedge sm(b) \leftrightarrow (st(b) \vee (i(a,b) \wedge sm(a))) \wedge sm(a) \leftrightarrow st(a)$$

- Rewrite in CNF (as usual)

Search-based

Inference in Bayesian Networks

$$P(E \mid g) = \frac{P(E \wedge g)}{\sum_E P(E \wedge g)}$$



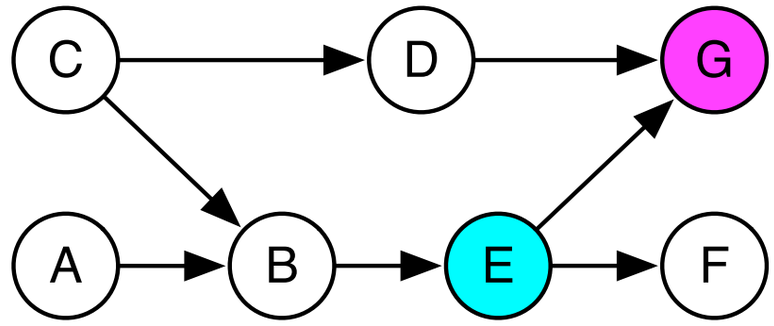
$$P(E \wedge g) = \sum_{ABCDF} P(ABCDEFg)$$

$$= \sum_F \sum_B \sum_C \sum_A \sum_D P(A)P(B \mid AC)P(C)P(D \mid C)$$
$$P(E \mid B)P(F \mid E)P(g \mid ED)$$

$$= \left(\sum_F P(F \mid E)\right)$$
$$\sum_B P(E \mid B) \sum_C \left(P(C)\left(\sum_A P(A)P(B \mid AC)\right)\right.$$
$$\left.\left(\sum_D P(D \mid C)P(g \mid ED)\right)\right)$$

Inference in Bayesian Networks

$$P(E \mid g) = \frac{P(E \wedge g)}{\sum_E P(E \wedge g)}$$



$$P(E \wedge g) = \sum_{ABCDEF} P(ABCDEFg)$$

SUM

$$= \sum_F \sum_B \sum_C \sum_A \sum_D P(A)P(B \mid AC)P(C)P(D \mid C)$$

PRODUCT

$$P(E \mid B)P(F \mid E)P(g \mid ED)$$

$$= \left(\sum_F P(F \mid E)\right)$$

$$\sum_B P(E \mid B) \sum_C \left(P(C) \left(\sum_A P(A)P(B \mid AC) \right) \right.$$

$$\left. \left(\sum_D P(D \mid C)P(g \mid ED) \right) \right)$$

SAT

Satisfiability of CNF formula

$$(A \vee B) \wedge (\neg B \vee C)$$

$$\bigvee_{A,B,C \in \{true,false\}} (A \vee B) \wedge (\neg B \vee C)$$

$$\bigvee_{A,B,C \in \{true,false\}} c_1(A,B) \wedge c_2(B,C)$$

$$\bigvee_{A,B,C \in \{true,false\}} \bigwedge_i c_i(A,B,C)$$

#SAT

Counting the number of satisfying assignments

$$\sum_{A,B,C \in \{true,false\}} true?(A \vee B) \times true?(B \vee C)$$

$$\sum_{A,B,C \in \{true,false\}} \prod_i c_i(A,B,C)$$

Weighted Model Counting

$$WMC(\phi) = \sum_{I_V \models \phi} \prod_{l \in I_V} w(l)$$

Weighted Model Counting

propositional formula in conjunctive normal form (CNF)

$$WMC(\phi) = \sum_{I_V \models \phi} \prod_{l \in I_V} w(l)$$

Weighted Model Counting

propositional formula in conjunctive normal form (CNF)

$$WMC(\phi) = \sum_{I_V \models \phi} \prod_{l \in I_V} w(l)$$

interpretations (truth value assignments) of propositional variables

Weighted Model Counting

propositional formula in conjunctive normal form (CNF)

$$WMC(\phi) = \sum_{I_V \models \phi} \prod_{l \in I_V} w(l)$$

interpretations (truth value assignments) of propositional variables

weight of literal

Weighted Model Counting

propositional formula in conjunctive normal form (CNF)

$$WMC(\phi) = \sum_{I_V \models \phi} \prod_{l \in I_V} w(l)$$

interpretations (truth value assignments) of propositional variables

possible worlds

weight of literal

Weighted Model Counting

propositional formula in conjunctive normal form (CNF)

$$WMC(\phi) = \sum_{I_V \models \phi} \prod_{l \in I_V} w(l)$$

interpretations (truth value assignments) of propositional variables

possible worlds

weight of literal

for instance …

w(f) = p

w(not f) = 1−p

Weighted Model Counting

$$WMC(\phi) = \sum_{I_V \models \phi} \prod_{l \in I_V} w(l)$$

$$WMC((A \vee B) \wedge (\neg A \vee \neg B)) =$$
$$w(A) \times w(\neg B) + w(B) \times w(\neg B)$$

Sum Product

$\text{SUMPRODUCT}\ (V, F, \oplus, \otimes)$ $\qquad \oplus_{X_1} \cdots \oplus_{X_m} \otimes_{i=1}^{m} f_i(V_i)$

$\oplus, \otimes$ typically commutative, associative,
and $\otimes$ distributes over $\oplus$,
0 identity element for $\oplus$ and annihilating elements for $\otimes$,
1 identity element for $\otimes$

Sum Product

$\text{SUMPRODUCT}\ (V, F, \oplus, \otimes)$ $\quad\quad$ $\oplus_{X_1} \cdots \oplus_{X_m} \otimes_{i=1}^{m} f_i(V_i)$

Variables

$\oplus, \otimes$ typically commutative, associative,
and $\otimes$ distributes over $\oplus$,
0 identity element for $\oplus$ and annihilating elements for $\otimes$,
1 identity element for $\otimes$

Sum Product

$$\text{SUMPRODUCT}\ (V, F, \oplus, \otimes) \qquad \oplus_{X_1} \cdots \oplus_{X_m} \otimes_{i=1}^{m} f_i(V_i)$$

Variables

Factors

$\oplus, \otimes$ typically commutative, associative,
and $\otimes$ distributes over $\oplus$,
0 identity element for $\oplus$ and annihilating elements for $\otimes$,
1 identity element for $\otimes$

Inference

- Many algorithms exist for many variants of SUMPRODUCT
- Variable elimination, recursive conditioning for Bayes;
- DPLL and variants for SAT and extensions
- AND/OR search tree …

DPLL

procedure $DPLL(Vars:$ variables, $Fs:$ set of factors):
    if $Fs = \{\}$ **return** 1
    else if $f \in Fs$ can be evaluated
        return $f \otimes DPLL(Vars, Fs - \{f\})$
    else select $v \in Vars$
        $Fs_T = Fs$ with v assigned $true$
        $Fs_F = Fs$ with v assigned $false$
        return $DPLL(V - \{v\}, Fs_T) \oplus DPLL(V - \{v\}, Fs_F)$

all variables assumed boolean (otherwise sum over all possible values *v* in last step)

DPLL

procedure $DPLL(Vars:$ variables, $Fs:$ set of factors):

if $Fs = \{\}$ **return** 1

else if $\{\} \in Fs$

return 0 (false)

else select $v \in Vars$

$Fs_T = Fs$ with v assigned $true$

$Fs_F = Fs$ with v assigned $false$

return $DPLL(V - \{v\}, Fs_T) \oplus DPLL(V - \{v\}, Fs_F)$

DPLL-SAT refinement

procedure $DPLL(Vars:$ variables, $Fs:$ set of factors):

if $Fs = \{\}$ **return** 1

else if $\{\} \in Fs$

return 0 (false)

else if there is a literal l in Fs so that $\bar{l}$ does not appear in Fs

$Fs' = Fs - \{f | f \in Fs$ and f contains $l\}$

return $DPLL(Vars - \{var(l)\}, Fs')$

else if Fs contains a unit clause l

$Fs' = \{f - \{\bar{l}\} | f \in Fs$ and f does not contain $l\}$

return $DPLL(Vars - \{var(l)\}, Fs')$

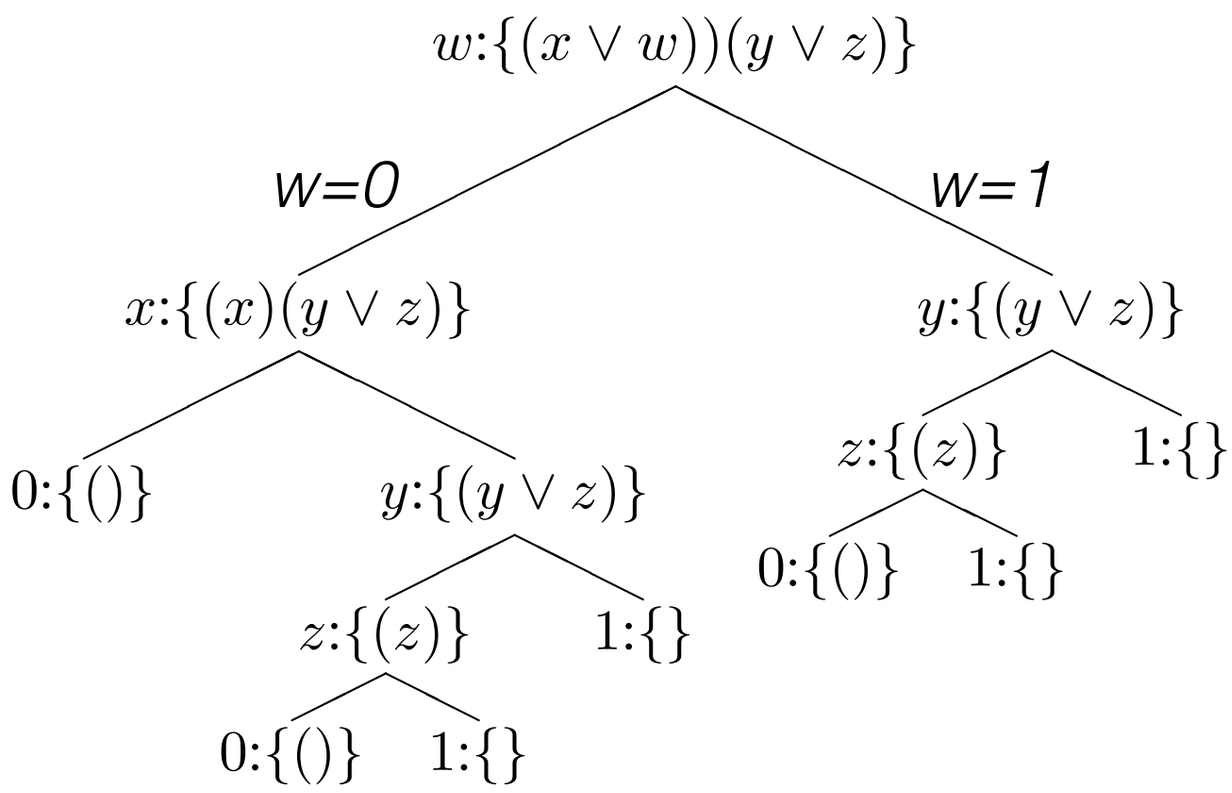
else select $v \in Vars$

$Fs_T = Fs$ with v assigned $true$

$Fs_F = Fs$ with v assigned $false$

return $DPLL(V - \{v\}, Fs_T) \oplus DPLL(V - \{v\}, Fs_F)$

Example DPLL

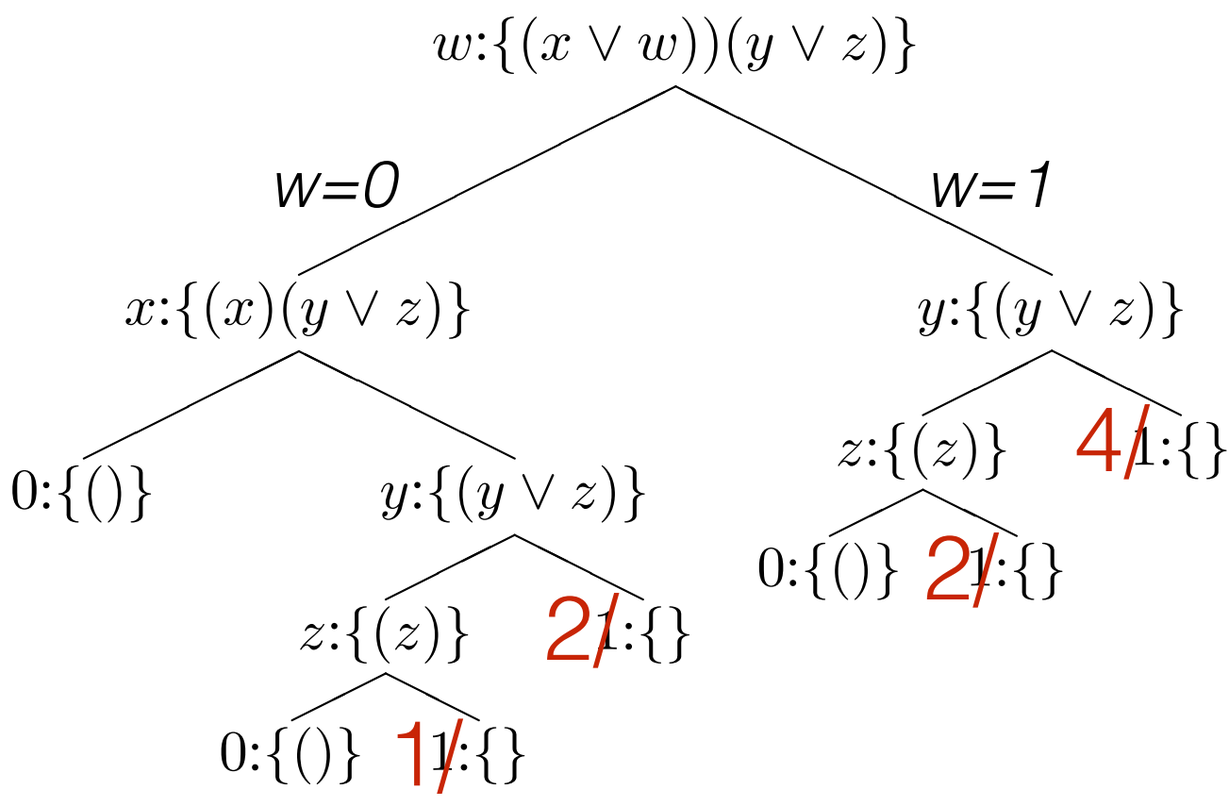


from Bacchus et al. JAIR 2009

#DPLL for #SAT

procedure $\#\ DPLL(Vars:$ variables, $Fs:$ set of factors):
    if $Fs = \{\}$ **return** $2^{|Vars|}$
    else if $\{\} \in Fs$
        return 0 (false)
    else select $v \in Vars$
        $Fs_T = Fs$ with v assigned $true$
        $Fs_F = Fs$ with v assigned $false$
        return $DPLL(V - \{v\}, Fs_T) \oplus DPLL(V - \{v\}, Fs_F)$

Example #DPLL

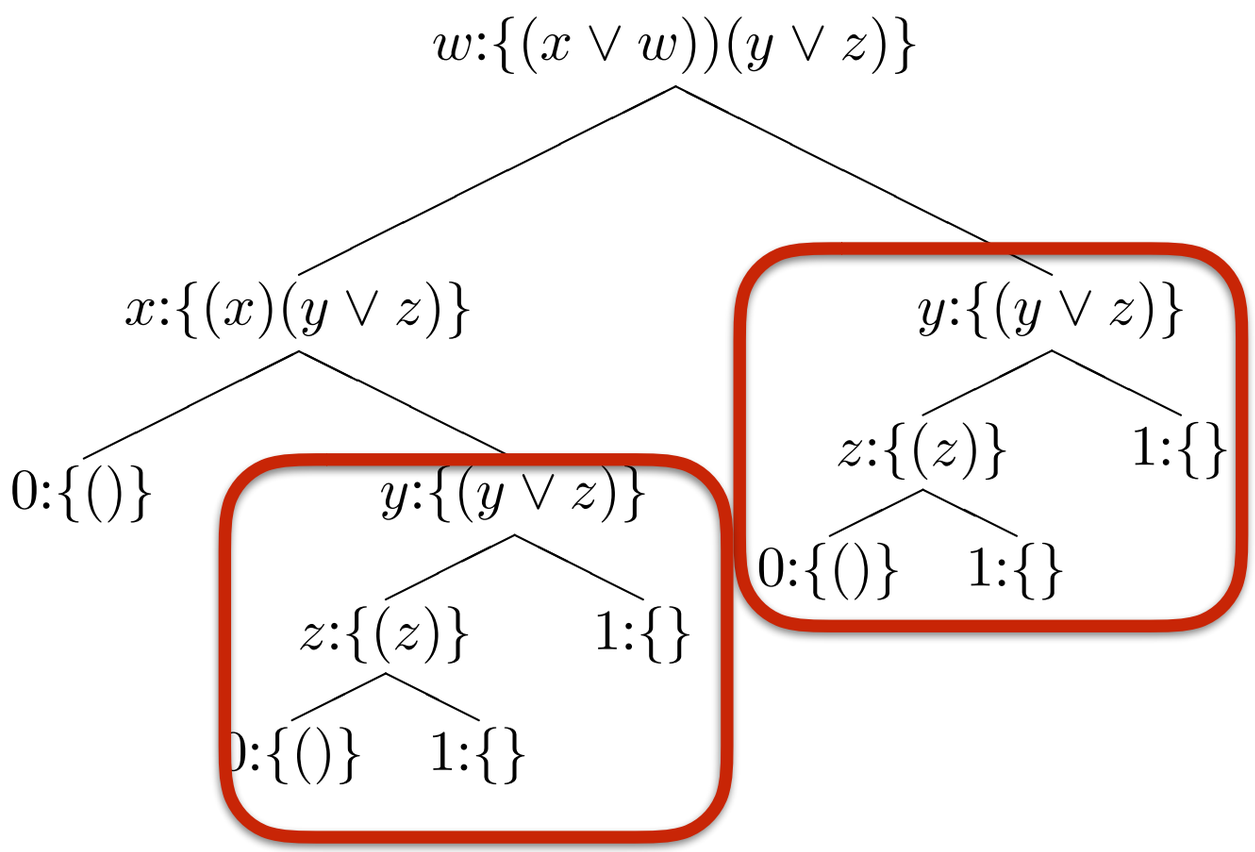


from Bacchus et al. JAIR 2009

#DPLL for #WMC

procedure $\#\ DPLL(Vars:$ variables, $Fs:$ set of factors, $w:$ weight funct
    if $Fs = \{\}$ **return** $\prod_{v \in Vars} w(v) + w(\neg v)$
    else if $\{\} \in Fs$
        return 0 (false)
    else select $v \in Vars$
        $Fs_T = Fs$ with v assigned $true$
        $Fs_F = Fs$ with v assigned $false$
        return $w(v) \otimes DPLL(V - \{v\}, Fs_T) \oplus w(\neg v) \otimes DPLL(V - \{v\},$

Example DPLL



from Bacchus et al. JAIR 2009

#DPLL-caching for #SAT

procedure $\#\ DPLL-cache(Vars:$ variables, $Fs:$ set of factors):
  if $\exists v$ such that $<< Vars, Fs >, val > \in cache$ **return** val
  $\%$ *also detect base cases*
  else select $v \in Vars$
  $Fs_T = Fs$ with v assigned $true$
  $Fs_F = Fs$ with v assigned $false$
  $val = DPLL(V - \{v\}, Fs_T) \oplus DPLL(V - \{v\}, Fs_F)$
  $cache \leftarrow cache \cup \{<< Vars, Fs >, val >\}$
  return val

there is also “component caching”,
caching at the clause level

Weighted Model Counting

$$WMC(\phi) = \sum_{I_V \models \phi} \prod_{l \in I_V} w(l)$$

Weighted Model Counting

propositional formula in conjunctive normal form (CNF)

$$WMC(\phi) = \sum_{I_V \models \phi} \prod_{l \in I_V} w(l)$$

Weighted Model Counting

propositional formula in conjunctive normal form (CNF)

$$WMC(\phi) = \sum_{I_V \models \phi} \prod_{l \in I_V} w(l)$$

interpretations (truth value assignments) of propositional variables

Weighted Model Counting

propositional formula in conjunctive normal form (CNF)

$$WMC(\phi) = \sum_{I_V \models \phi} \prod_{l \in I_V} w(l)$$

interpretations (truth value assignments) of propositional variables

weight of literal

Weighted Model Counting

propositional formula in conjunctive normal form (CNF)

given by ProbLog program & query

$$WMC(\phi) = \sum_{I_V \models \phi} \prod_{l \in I_V} w(l)$$

interpretations (truth value assignments) of propositional variables

weight of literal

Weighted Model Counting

propositional formula in conjunctive normal form (CNF)

given by ProbLog program & query

$$WMC(\phi) = \sum_{I_V \models \phi} \prod_{l \in I_V} w(l)$$

interpretations (truth value assignments) of propositional variables

possible worlds

weight of literal

Weighted Model Counting

propositional formula in conjunctive normal form (CNF)

given by ProbLog program & query

$$WMC(\phi) = \sum_{I_V \models \phi} \prod_{l \in I_V} w(l)$$

interpretations (truth value assignments) of propositional variables

possible worlds

weight of literal

for p::f,
w(f) = p
w(not f) = 1−p

Weighte

$$P(Q) = \sum_{F \cup R \models Q} \prod_{f \in F} p(f) \prod_{f \notin F} 1 - p(f)$$

propositional formula in conjunctive normal form (CNF)

given by ProbLog program & query

$$WMC(\phi) = \sum_{I_V \models \phi} \prod_{l \in I_V} w(l)$$

interpretations (truth value assignments) of propositional variables

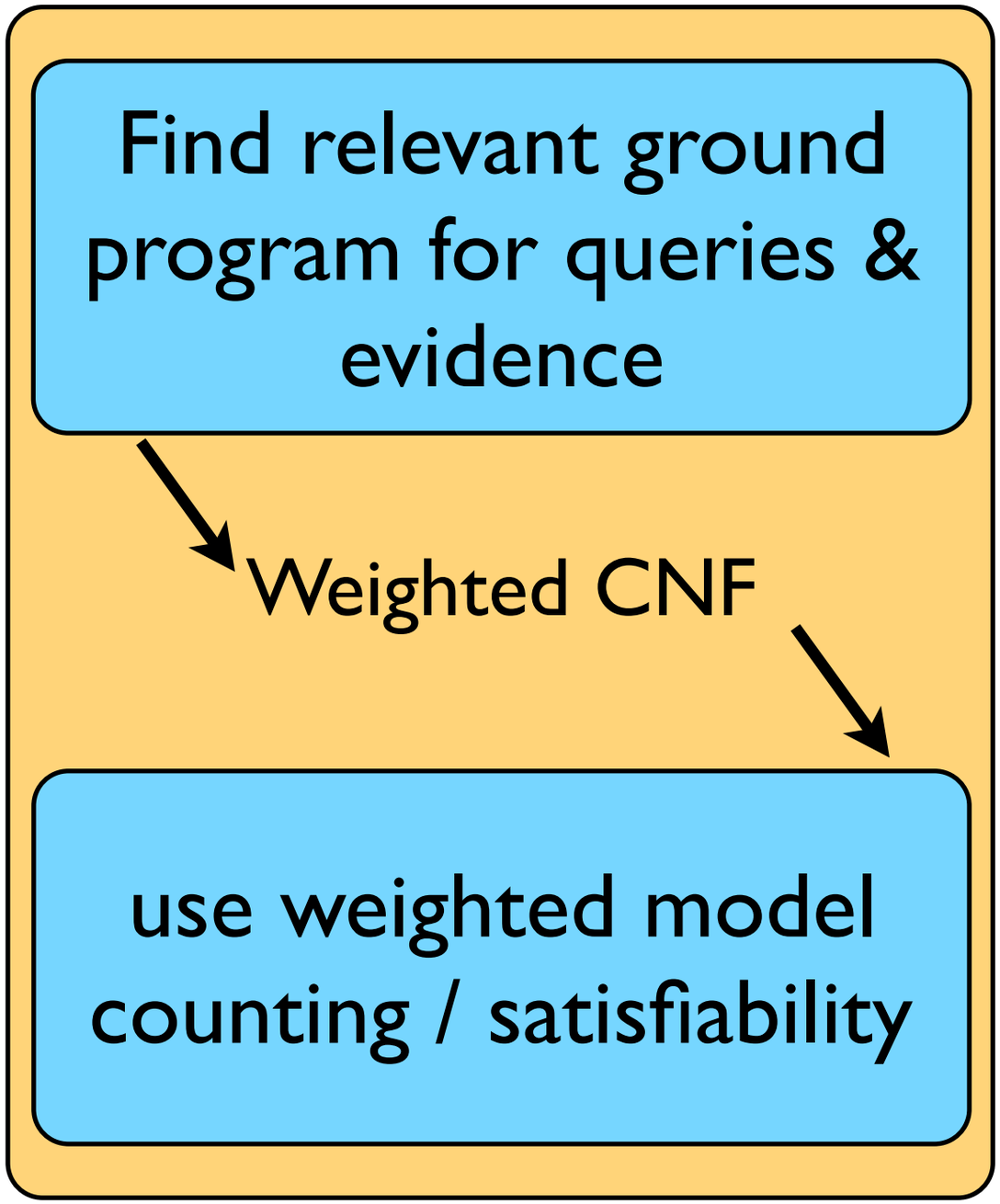
possible worlds

weight of literal

for p::f,
w(f) = p
w(not f) = 1−p

Current Approach

(ProbLog2)

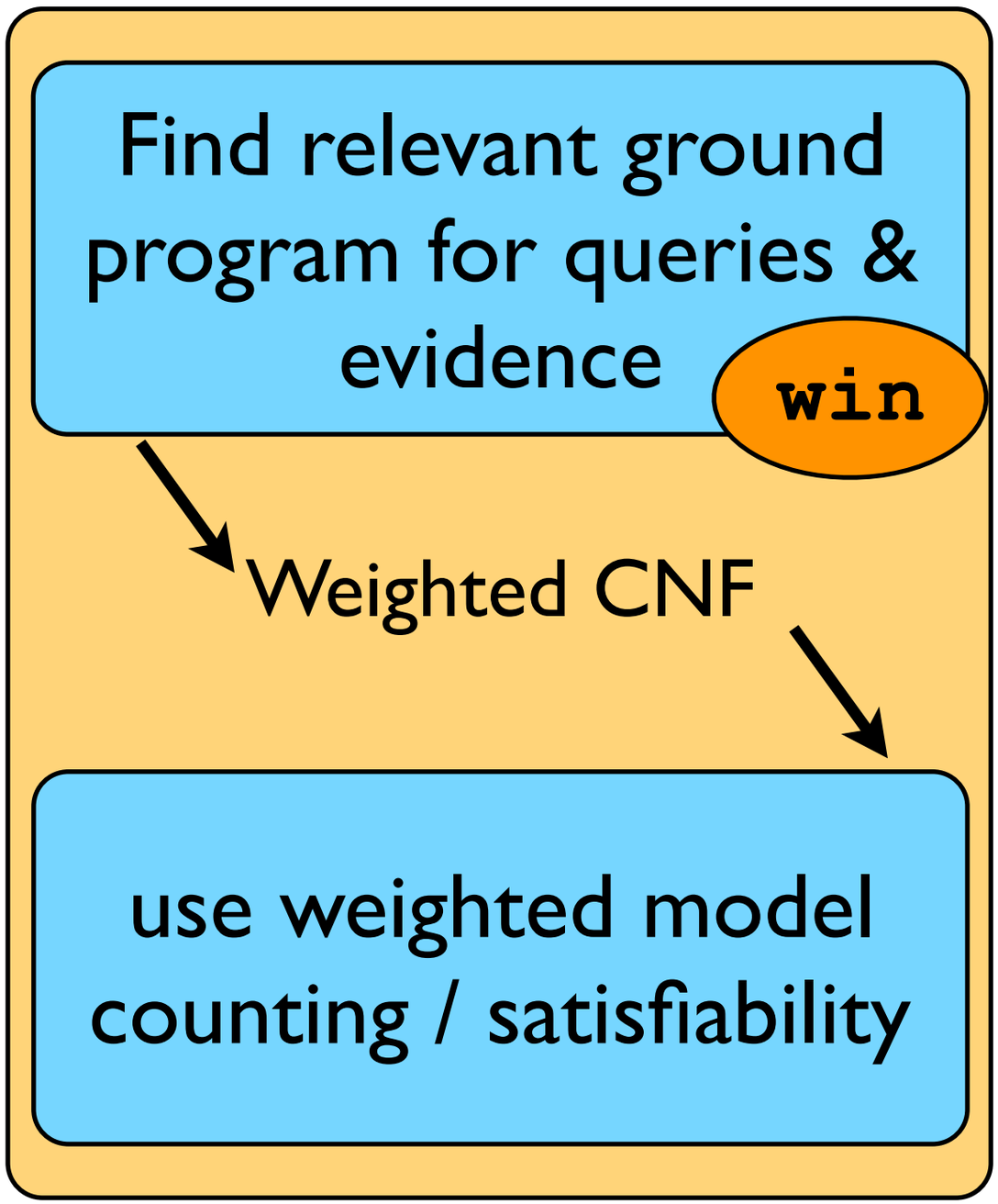


[Fierens et al, TPLP 14]

Current Approach

(ProbLog2)

```
0.4::heads(1).
0.7::heads(2).
0.5::heads(3).
win :- heads(1).
win :- heads(2),
       heads(3).
```

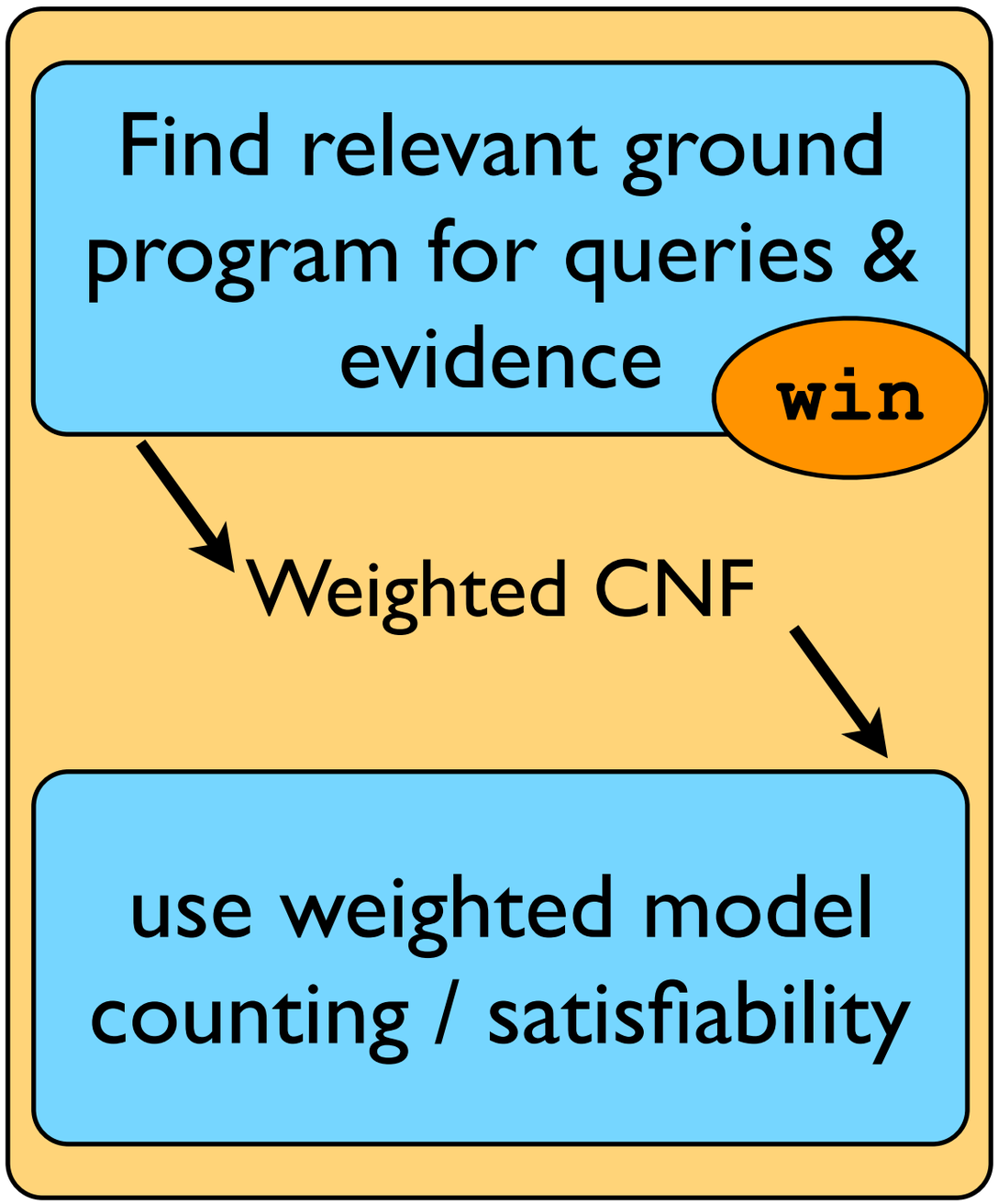


[Fierens et al, TPLP 14]

Current Approach

(ProbLog2)

```
0.4::heads(1).
0.7::heads(2).
0.5::heads(3).
win :- heads(1).
win :- heads(2),
       heads(3).
```

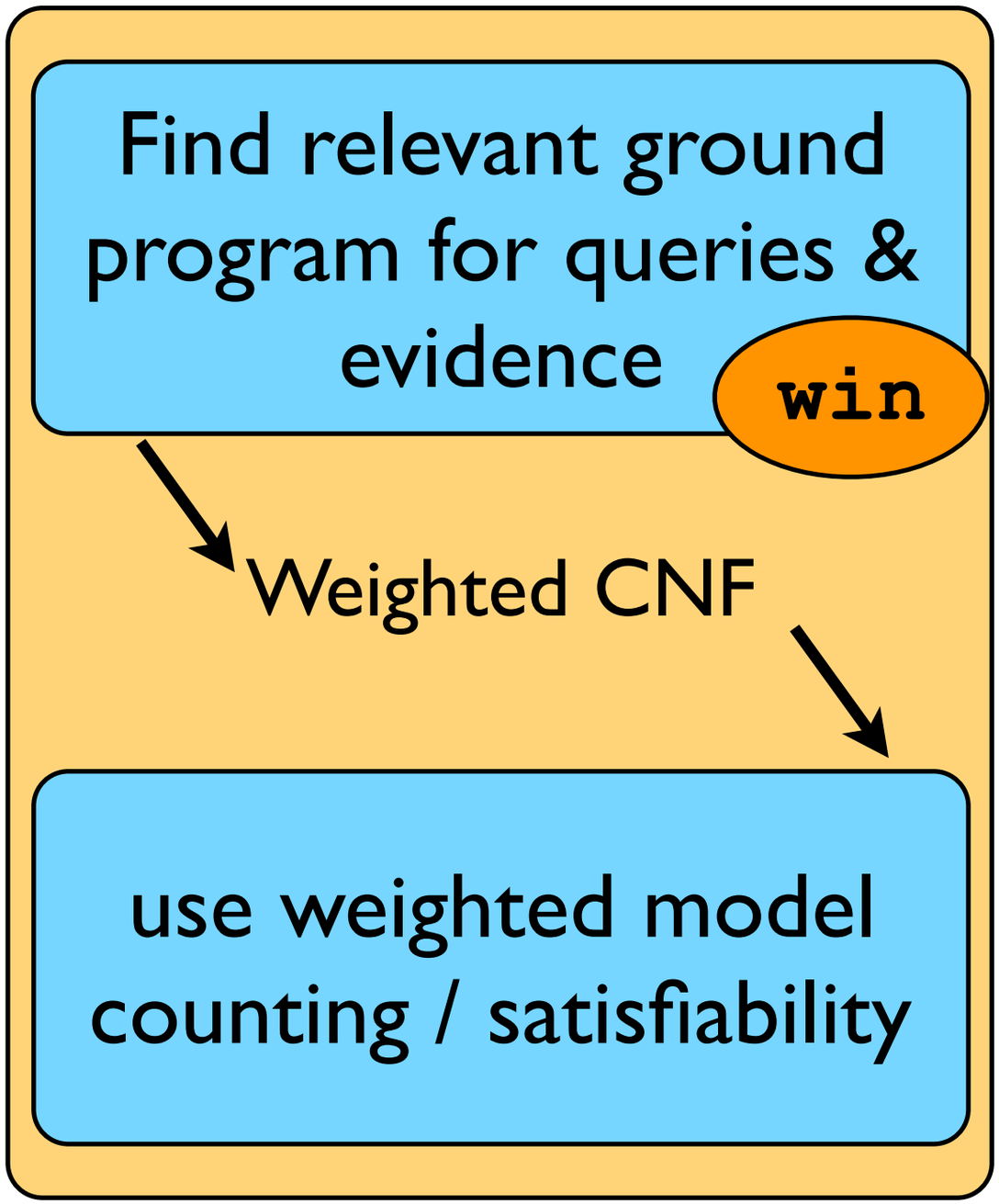



```
win :- heads(1).
win :- heads(2), heads(3).
```


[Fierens et al, TPLP 14]

Current Approach
(ProbLog2)

```
0.4::heads(1).
0.7::heads(2).
0.5::heads(3).
win :- heads(1).
win :- heads(2),
       heads(3).
```




```
win :- heads(1).
win :- heads(2), heads(3).
```

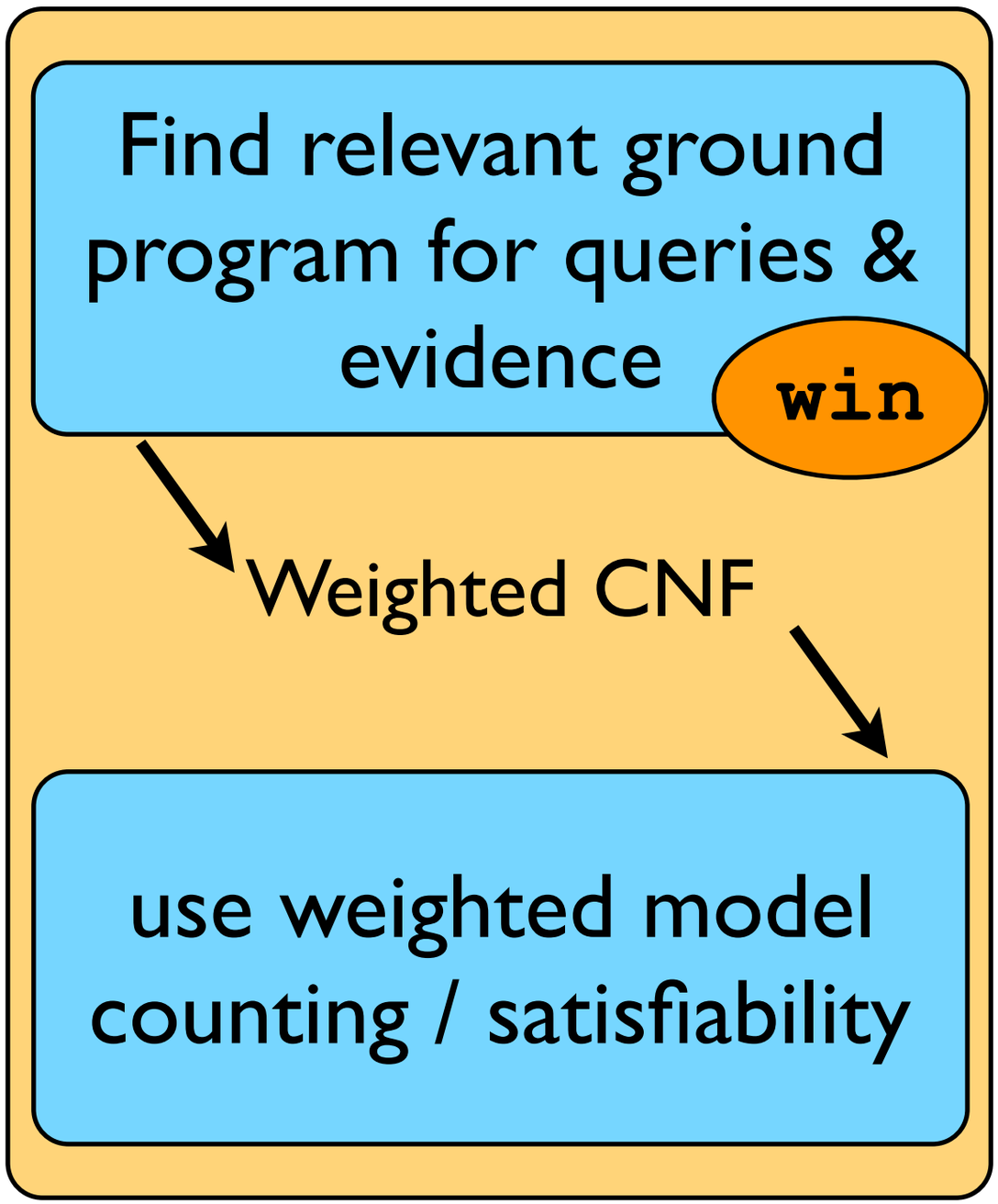
$$\text{win} \leftrightarrow h(1) \vee (h(2) \wedge h(3))$$

[Fierens et al, TPLP 14]

Current Approach

(ProbLog2)

```
0.4::heads(1).
0.7::heads(2).
0.5::heads(3).
win :- heads(1).
win :- heads(2),
       heads(3).
```




```
win :- heads(1).
win :- heads(2), heads(3).
```

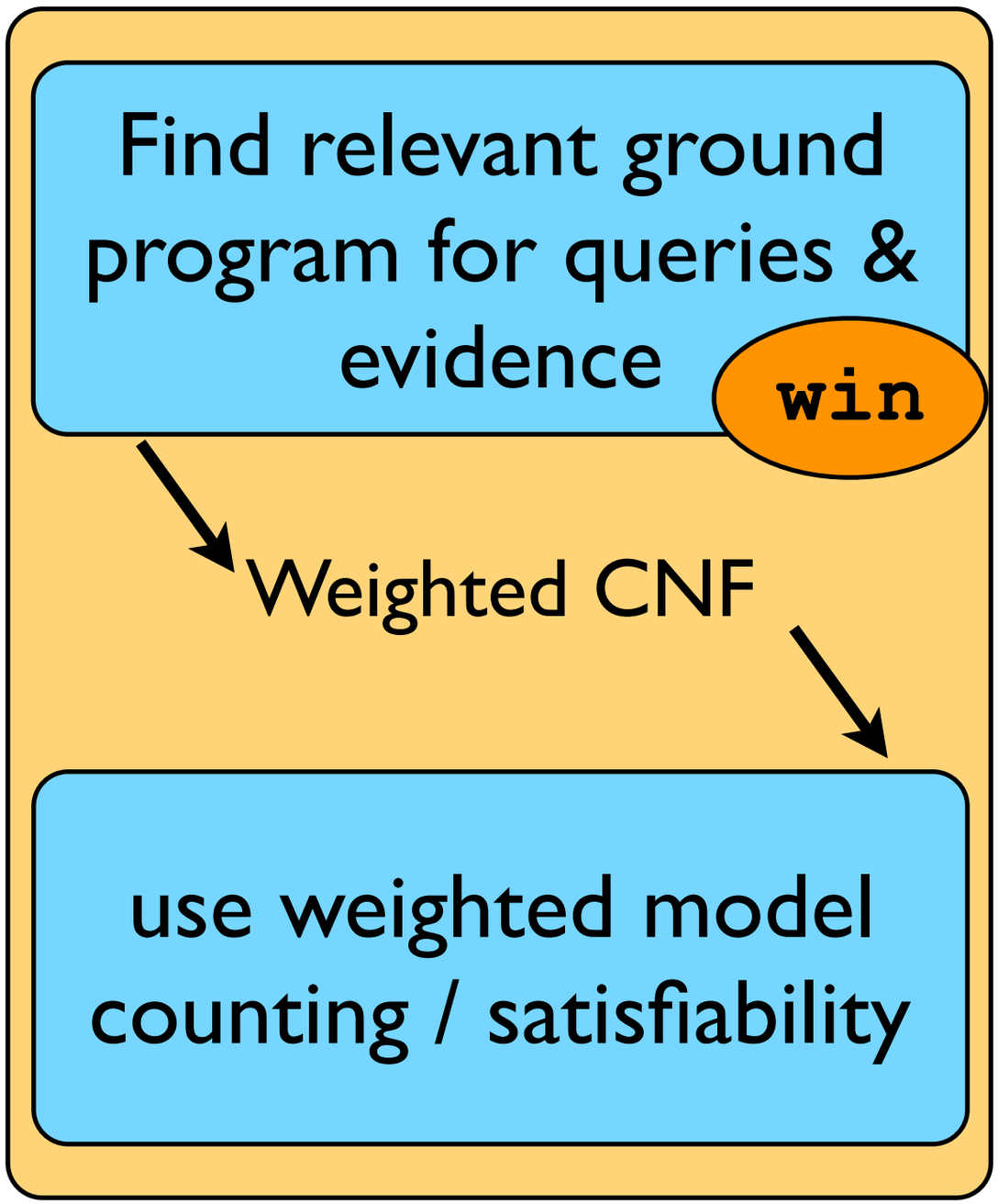
$$\text{win} \leftrightarrow h(1) \vee (h(2) \wedge h(3))$$

$$(\neg\text{win} \vee h(1) \vee h(2)) \wedge (\neg\text{win} \vee h(1) \vee h(3)) \wedge (\text{win} \vee \neg h(1)) \wedge (\text{win} \vee \neg h(2) \vee \neg h(3))$$

[Fierens et al, TPLP 14]

Current Approach
(ProbLog2)

```
0.4::heads(1).
0.7::heads(2).
0.5::heads(3).
win :- heads(1).
win :- heads(2),
       heads(3).
```




```
win :- heads(1).
win :- heads(2), heads(3).
```

$$\text{win} \leftrightarrow h(1) \vee (h(2) \wedge h(3))$$

$$(\neg\text{win} \vee h(1) \vee h(2)) \wedge (\neg\text{win} \vee h(1) \vee h(3)) \wedge (\text{win} \vee \neg h(1)) \wedge (\text{win} \vee \neg h(2) \vee \neg h(3))$$

$h(1) \rightarrow 0.4$	$h(2) \rightarrow 0.7$	$h(3) \rightarrow 0.5$
$\neg h(1) \rightarrow 0.6$	$\neg h(2) \rightarrow 0.3$	$\neg h(3) \rightarrow 0.5$

[Fierens et al, TPLP 14]

Current Approach
(ProbLog2)

```
0.4::heads(1).
0.7::heads(2).
0.5::heads(3).
win :- heads(1).
win :- heads(2),
       heads(3).
```

Find relevant ground program for queries & evidence

win

Weighted CNF

use weighted model counting / satisfiability

```
win :- heads(1).
win :- heads(2), heads(3).
```

$$win \leftrightarrow h(1) \vee (h(2) \wedge h(3))$$

$$(\neg win \vee h(1) \vee h(2)) \wedge (\neg win \vee h(1) \vee h(3)) \wedge (win \vee \neg h(1)) \wedge (win \vee \neg h(2) \vee \neg h(3))$$

use standard solver

$h(1) \rightarrow 0.4$ $h(2) \rightarrow 0.7$ $h(3) \rightarrow 0.5$

$\neg h(1) \rightarrow 0.6$ $\neg h(2) \rightarrow 0.3$ $\neg h(3) \rightarrow 0.5$

[Fierens et al, TPLP 14]

ProbLog Inference

- reduction to propositional formula
- addresses disjoint-sum-problem
- **but**: not all probabilistic logic programs face this problem! e.g., weather
- more generally: mutually exclusive proofs as assumed in PRISM

Algebraic ProbLog (semirings)

Dyna (Eisner et al.)

word(john,0,1), word(loves,1,2), word(Mary,2,3)
0.003:: np -> Mary as rewrite(np,Mary)=0.003
0.5::vp -> verb, np as rewrite(vp,verb,np)=0.5

- CKY Algorithm in Dyna

```
constit(X,I,J) += rewrite(X,W) * word(W,I,J).
constit(X,I,K) += rewrite(X,Y,Z) * constit(Y,I,J) * constit(Z,J,K).
goal += constit(S,0,N) * end(N).
```

$\sum_{W}$

$\sum_{Y,Z,J}$

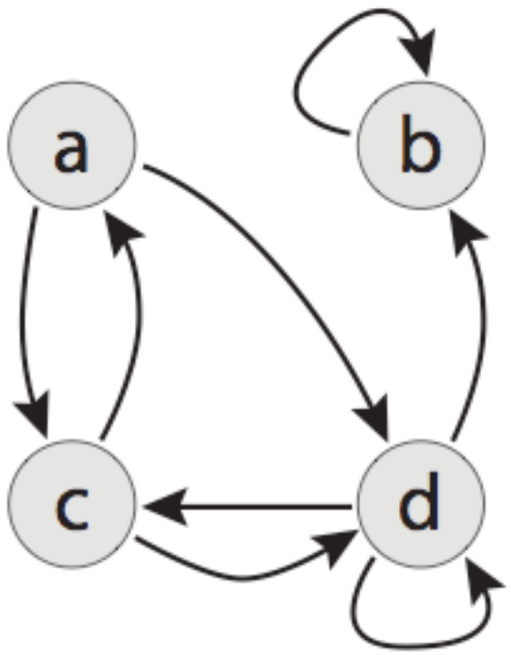
Origins in Natural Language Processing Context
Based on Dynamic Programming
Useful for SRL/NLP/Programming

Dyna

- Weighted Logic Programs (but not Prolog)
- Inspired by NLP
- Arbitrary semiring weights
- Forward reasoning

```
reachable(Q) :- initial(Q).
reachable(Q) :- reachable(P), edge(P, Q).
```

Fig. 1. A simple bottom-up logic program for graph reachability



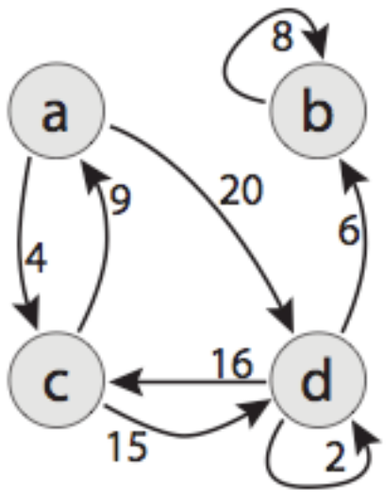

```
initial(a) = T   edge(c, d) = T
 edge(a, c) = T   edge(d, b) = T
 edge(a, d) = T   edge(d, c) = T
 edge(b, b) = T   edge(d, d) = T
 edge(c, a) = T
```

Fig. 2. A directed graph and the corresponding initial database

[Figures: Cohen et al, ICLP 08]

$$\texttt{reachable(Q)} \oplus\!= \texttt{initial(Q)}.$$

$$\texttt{reachable(Q)} \oplus\!= \texttt{reachable(P)} \otimes \texttt{edge(P, Q)}.$$



$$\begin{aligned}
\texttt{initial(a)} &= 0 & \texttt{edge(c, d)} &= 15\\
\texttt{edge(a, c)} &= 4 & \texttt{edge(d, b)} &= 6\\
\texttt{edge(a, d)} &= 20 & \texttt{edge(d, c)} &= 16\\
\texttt{edge(b, b)} &= 8 & \texttt{edge(d, d)} &= 2\\
\texttt{edge(c, a)} &= 9
\end{aligned}$$

Fig. 3. A cost graph and the corresponding initial database

[Figures: Cohen et al, ICLP 08]

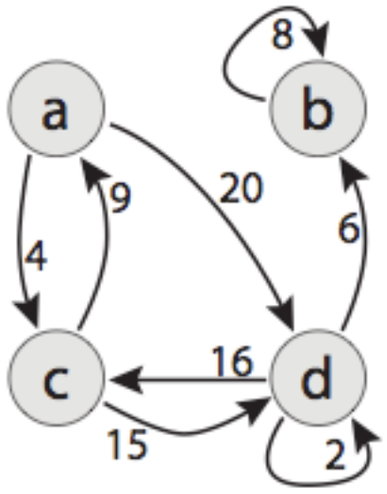
$$\texttt{reachable}(\texttt{Q}) \oplus\!= \texttt{initial}(\texttt{Q}).$$

$$\texttt{reachable}(\texttt{Q}) \oplus\!= \texttt{reachable}(\texttt{P}) \otimes \texttt{edge}(\texttt{P}, \texttt{Q}).$$

$\oplus \quad \rightarrow \quad \min$

$\otimes \quad \rightarrow \quad +$

shortest path



$$\texttt{initial}(\texttt{a}) = 0 \qquad \texttt{edge}(\texttt{c}, \texttt{d}) = 15$$
$$\texttt{edge}(\texttt{a}, \texttt{c}) = 4 \qquad \texttt{edge}(\texttt{d}, \texttt{b}) = 6$$
$$\texttt{edge}(\texttt{a}, \texttt{d}) = 20 \qquad \texttt{edge}(\texttt{d}, \texttt{c}) = 16$$
$$\texttt{edge}(\texttt{b}, \texttt{b}) = 8 \qquad \texttt{edge}(\texttt{d}, \texttt{d}) = 2$$
$$\texttt{edge}(\texttt{c}, \texttt{a}) = 9$$

Fig. 3. A cost graph and the corresponding initial database

[Figures: Cohen et al, ICLP 08]

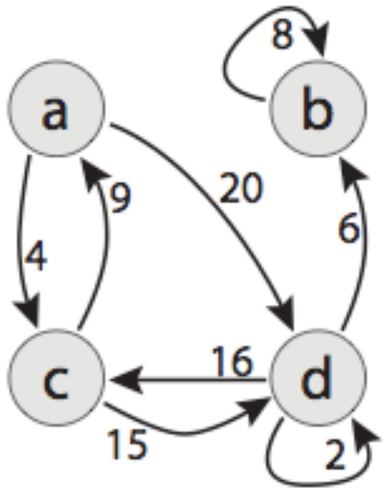
$$\texttt{reachable}(\texttt{Q}) \oplus\!= \texttt{initial}(\texttt{Q}).$$

$$\texttt{reachable}(\texttt{Q}) \oplus\!= \texttt{reachable}(\texttt{P}) \otimes \texttt{edge}(\texttt{P}, \texttt{Q}).$$

$\oplus \rightarrow \min$

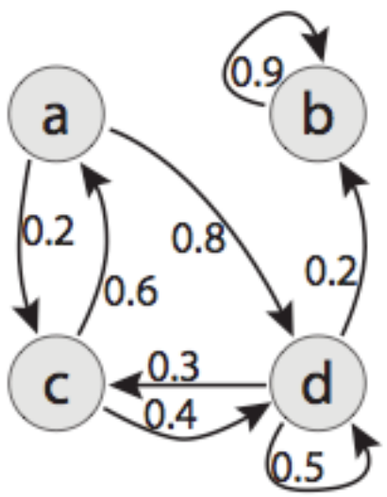
$\otimes \rightarrow +$

shortest path



$$\texttt{initial}(\texttt{a}) = 0 \quad \texttt{edge}(\texttt{c},\texttt{d}) = 15$$
$$\texttt{edge}(\texttt{a},\texttt{c}) = 4 \quad \texttt{edge}(\texttt{d},\texttt{b}) = 6$$
$$\texttt{edge}(\texttt{a},\texttt{d}) = 20 \quad \texttt{edge}(\texttt{d},\texttt{c}) = 16$$
$$\texttt{edge}(\texttt{b},\texttt{b}) = 8 \quad \texttt{edge}(\texttt{d},\texttt{d}) = 2$$
$$\texttt{edge}(\texttt{c},\texttt{a}) = 9$$

Fig. 3. A cost graph and the corresponding initial database



$$\texttt{initial}(\texttt{a}) = 1 \quad \texttt{edge}(\texttt{c},\texttt{d}) = 0.4$$
$$\texttt{edge}(\texttt{a},\texttt{c}) = 0.2 \quad \texttt{edge}(\texttt{d},\texttt{b}) = 0.2$$
$$\texttt{edge}(\texttt{a},\texttt{d}) = 0.8 \quad \texttt{edge}(\texttt{d},\texttt{c}) = 0.3$$
$$\texttt{edge}(\texttt{b},\texttt{b}) = 0.9 \quad \texttt{edge}(\texttt{d},\texttt{d}) = 0.5$$
$$\texttt{edge}(\texttt{c},\texttt{a}) = 0.6$$

Fig. 4. A probabilistic graph and the corresponding initial database. With stopping probabilities made explicit, this would encode a Markov model.

[Figures: Cohen et al, ICLP 08]

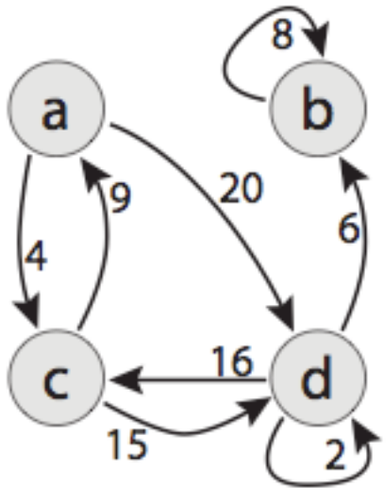
$$\texttt{reachable}(\texttt{Q}) \oplus\!= \texttt{initial}(\texttt{Q}).$$

$$\texttt{reachable}(\texttt{Q}) \oplus\!= \texttt{reachable}(\texttt{P}) \otimes \texttt{edge}(\texttt{P}, \texttt{Q}).$$

$\oplus \rightarrow \min$

$\otimes \rightarrow +$

shortest path



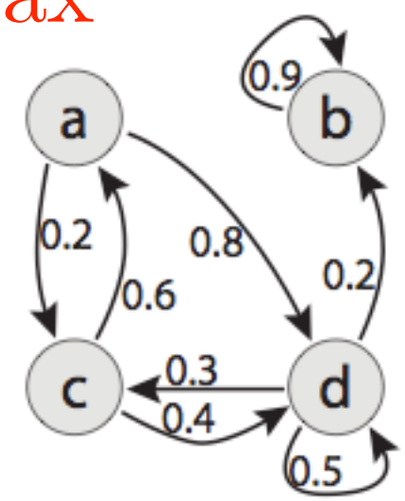
$$\texttt{initial}(\texttt{a}) = 0 \quad \texttt{edge}(\texttt{c},\texttt{d}) = 15$$
$$\texttt{edge}(\texttt{a},\texttt{c}) = 4 \quad \texttt{edge}(\texttt{d},\texttt{b}) = 6$$
$$\texttt{edge}(\texttt{a},\texttt{d}) = 20 \quad \texttt{edge}(\texttt{d},\texttt{c}) = 16$$
$$\texttt{edge}(\texttt{b},\texttt{b}) = 8 \quad \texttt{edge}(\texttt{d},\texttt{d}) = 2$$
$$\texttt{edge}(\texttt{c},\texttt{a}) = 9$$

Fig. 3. A cost graph and the corresponding initial database

$\oplus \rightarrow \max$

$\otimes \rightarrow \cdot$

most likely path



$$\texttt{initial}(\texttt{a}) = 1 \quad \texttt{edge}(\texttt{c},\texttt{d}) = 0.4$$
$$\texttt{edge}(\texttt{a},\texttt{c}) = 0.2 \quad \texttt{edge}(\texttt{d},\texttt{b}) = 0.2$$
$$\texttt{edge}(\texttt{a},\texttt{d}) = 0.8 \quad \texttt{edge}(\texttt{d},\texttt{c}) = 0.3$$
$$\texttt{edge}(\texttt{b},\texttt{b}) = 0.9 \quad \texttt{edge}(\texttt{d},\texttt{d}) = 0.5$$
$$\texttt{edge}(\texttt{c},\texttt{a}) = 0.6$$

Fig. 4. A probabilistic graph and the corresponding initial database. With stopping probabilities made explicit, this would encode a Markov model.

[Figures: Cohen et al, ICLP 08]

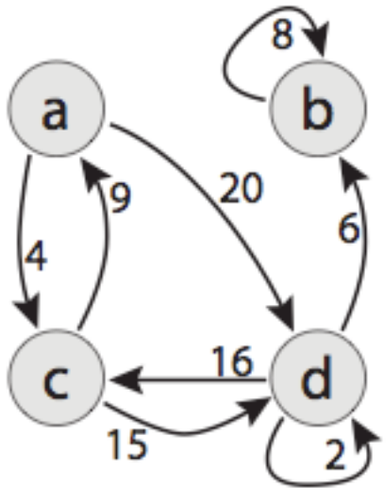
$$\texttt{reachable}(\texttt{Q}) \oplus\!= \texttt{initial}(\texttt{Q}).$$

$$\texttt{reachable}(\texttt{Q}) \oplus\!= \texttt{reachable}(\texttt{P}) \otimes \texttt{edge}(\texttt{P}, \texttt{Q}).$$

$\oplus \rightarrow \min$

$\otimes \rightarrow +$

shortest path



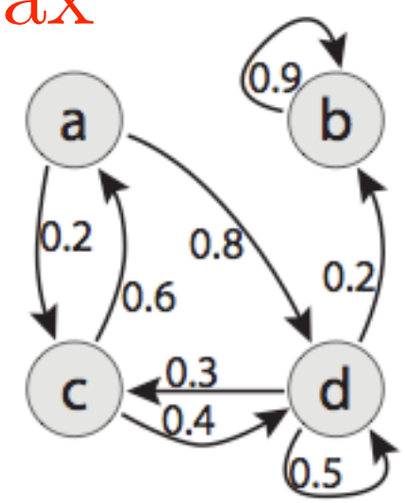
$$\texttt{initial}(\texttt{a}) = 0 \quad \texttt{edge}(\texttt{c},\texttt{d}) = 15$$
$$\texttt{edge}(\texttt{a},\texttt{c}) = 4 \quad \texttt{edge}(\texttt{d},\texttt{b}) = 6$$
$$\texttt{edge}(\texttt{a},\texttt{d}) = 20 \quad \texttt{edge}(\texttt{d},\texttt{c}) = 16$$
$$\texttt{edge}(\texttt{b},\texttt{b}) = 8 \quad \texttt{edge}(\texttt{d},\texttt{d}) = 2$$
$$\texttt{edge}(\texttt{c},\texttt{a}) = 9$$

Fig. 3. A cost graph and the corresponding initial database

$\oplus \rightarrow \max$

$\otimes \rightarrow \cdot$

most likely path



$$\texttt{initial}(\texttt{a}) = 1 \quad \texttt{edge}(\texttt{c},\texttt{d}) = 0.4$$
$$\texttt{edge}(\texttt{a},\texttt{c}) = 0.2 \quad \texttt{edge}(\texttt{d},\texttt{b}) = 0.2$$
$$\texttt{edge}(\texttt{a},\texttt{d}) = 0.8 \quad \texttt{edge}(\texttt{d},\texttt{c}) = 0.3$$
$$\texttt{edge}(\texttt{b},\texttt{b}) = 0.9 \quad \texttt{edge}(\texttt{d},\texttt{d}) = 0.5$$
$$\texttt{edge}(\texttt{c},\texttt{a}) = 0.6$$

$\oplus \rightarrow +$

$\otimes \rightarrow \cdot$

PRISM

Fig. 4. A probabilistic graph and the corresponding initial database. With stopping probabilities made explicit, this would encode a Markov model.

[Figures: Cohen et al, ICLP 08]

aProbLog: algebraic Prolog

[Kimmig et al, AAAI 11]

aProbLog: algebraic Prolog

[Kimmig et al, AAAI 11]

- Prolog + labeled facts

aProbLog: algebraic Prolog

[Kimmig et al, AAAI 11]

- Prolog + labeled facts
- labels from commutative semiring
  (e.g. probabilities, costs, polynomials,
  Boolean functions, datastructures, ...)

aProbLog: algebraic Prolog

[Kimmig et al, AAAI 11]

- Prolog + labeled facts
- labels from commutative semiring
  (e.g. probabilities, costs, polynomials,
  Boolean functions, datastructures, ...)
- inference based on ProbLog principles

aProbLog: algebraic Prolog

[Kimmig et al, AAAI 11]

- Prolog + labeled facts
- labels from commutative semiring
  (e.g. probabilities, costs, polynomials,
  Boolean functions, datastructures, ...)
- inference based on ProbLog principles

	neutral sum	NSP
disjoint sum	**SAT**	**MPE**
DSP	**PROB**	**#SAT**

aProbLog: algebraic Prolog

[Kimmig et al, AAAI 11]

- Prolog + labeled facts
- labels from commutative semiring (e.g. probabilities, costs, polynomials, Boolean functions, datastructures, ...)
- inference based on ProbLog principles

one world - several proofs

	neutral sum	NSP
disjoint sum	**SAT**	**MPE**
DSP	**PROB**	**#SAT**

aProbLog: algebraic Prolog

[Kimmig et al, AAAI 11]

- Prolog + labeled facts
- labels from commutative semiring
  (e.g. probabilities, costs, polynomials,
  Boolean functions, datastructures, ...)
- inference based on ProbLog principles

one proof - several worlds

one world - several proofs

	neutral sum	NSP
disjoint sum	**SAT**	**MPE**
DSP	**PROB**	**#SAT**

Algebraic Prolog (aProbLog)

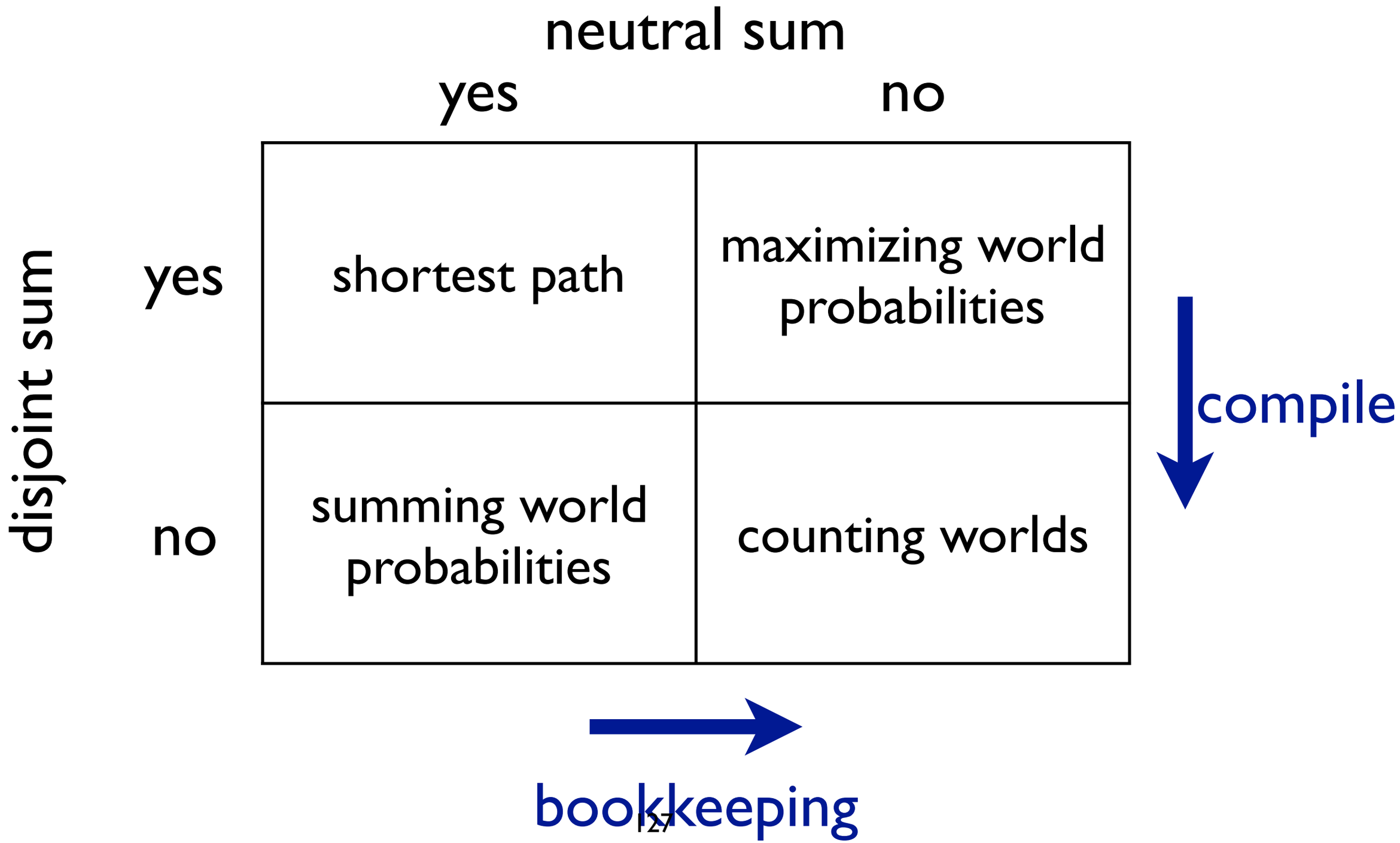
- commutative semiring $(A,\oplus,\otimes,e^{\oplus},e^{\otimes})$
- algebraic ground literals
  $L(F) = \{f_1,...,f_n\} \cup \{\neg f_1,...,\neg f_n\}$
- background knowledge clauses
- labeling function $\alpha: L(F) \rightarrow A$

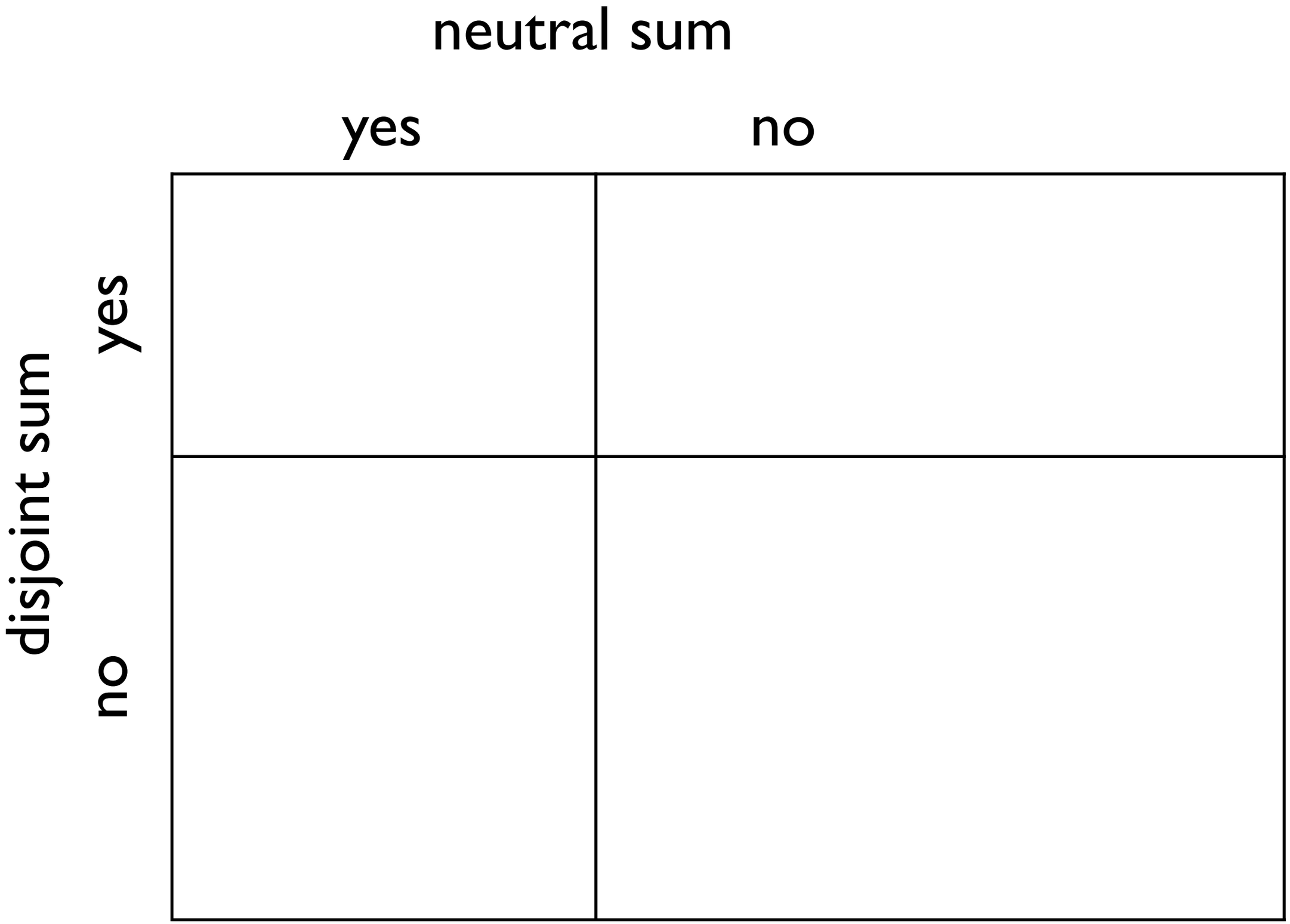
Algebraic Prolog (aProbLog)

- commutative semiring $(A,\oplus,\otimes,e^{\oplus},e^{\otimes})$
- algebraic ground literals
  $L(F) = \{f_1,...,f_n\} \cup \{\neg f_1,...,\neg f_n\}$
- background knowledge clauses
- labeling function $\alpha: L(F) \rightarrow A$

$$L(\text{query}) = \bigoplus_{\text{worlds}} \bigotimes_{\text{literals}} \alpha(l)$$

Inference settings




neutral sum
yes
no
disjoint sum
yes
no

neutral sum

min-sum

disjoint sum	yes	no
yes	$(a \wedge b) \vee (a \wedge c)$	
no		

neutral sum

min-sum yes no *most likely*

disjoint sum	yes	no
yes	$(a \wedge b)$ $\vee(a \wedge c)$	$(a \wedge b \wedge (c \vee \neg c))$ $\vee(a \wedge c \wedge (b \vee \neg b))$
no		

expand

neutral sum

min-sum yes no *most likely*

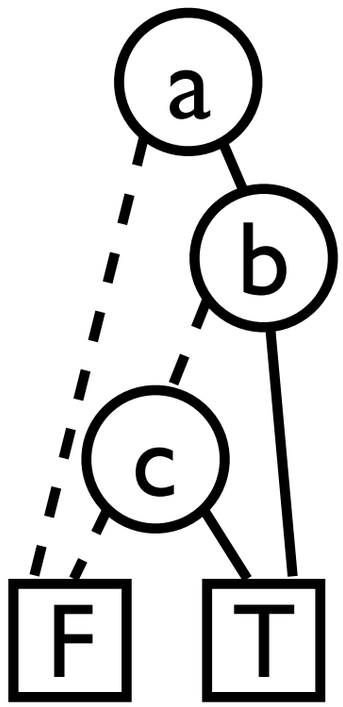
disjoint sum

yes

$(a \wedge b)$
$\vee(a \wedge c)$

$(a \wedge b \wedge (c \vee \neg c))$
$\vee(a \wedge c \wedge (b \vee \neg b))$

no



filter

probability

expand

neutral sum

min-sum

yes

no

most likely

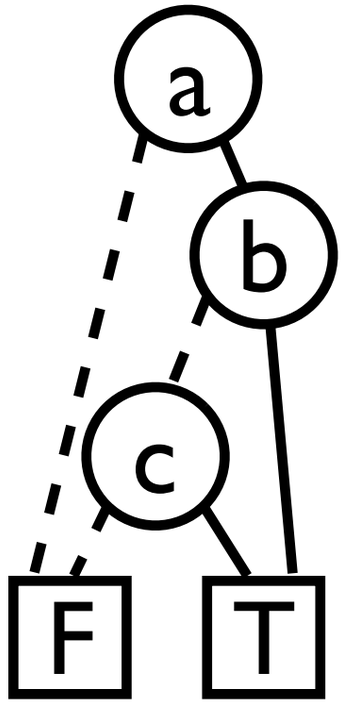
disjoint sum

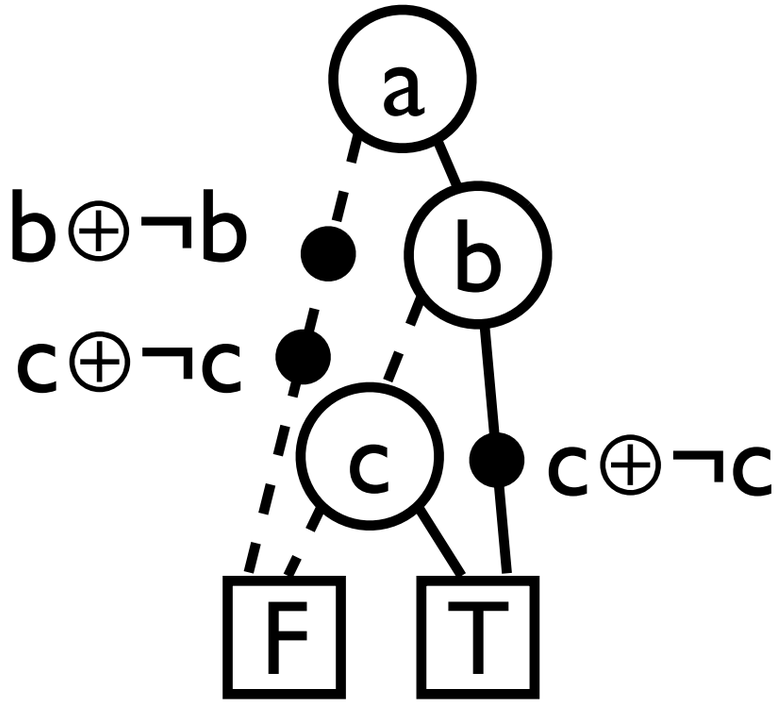
yes

no

$(a \wedge b)$
$\vee(a \wedge c)$

$(a \wedge b \wedge (c \vee \neg c))$
$\vee(a \wedge c \wedge (b \vee \neg b))$





filter

probability

expand

expected cost

Approximate Inference

Approximate Inference

- Lower and upper bounds

$$\phi_L \models \phi \models \phi_U$$

$$P(\phi_L) \leq P(\phi) \leq P(\phi_U)$$

Approximate Inference

- Lower and upper bounds

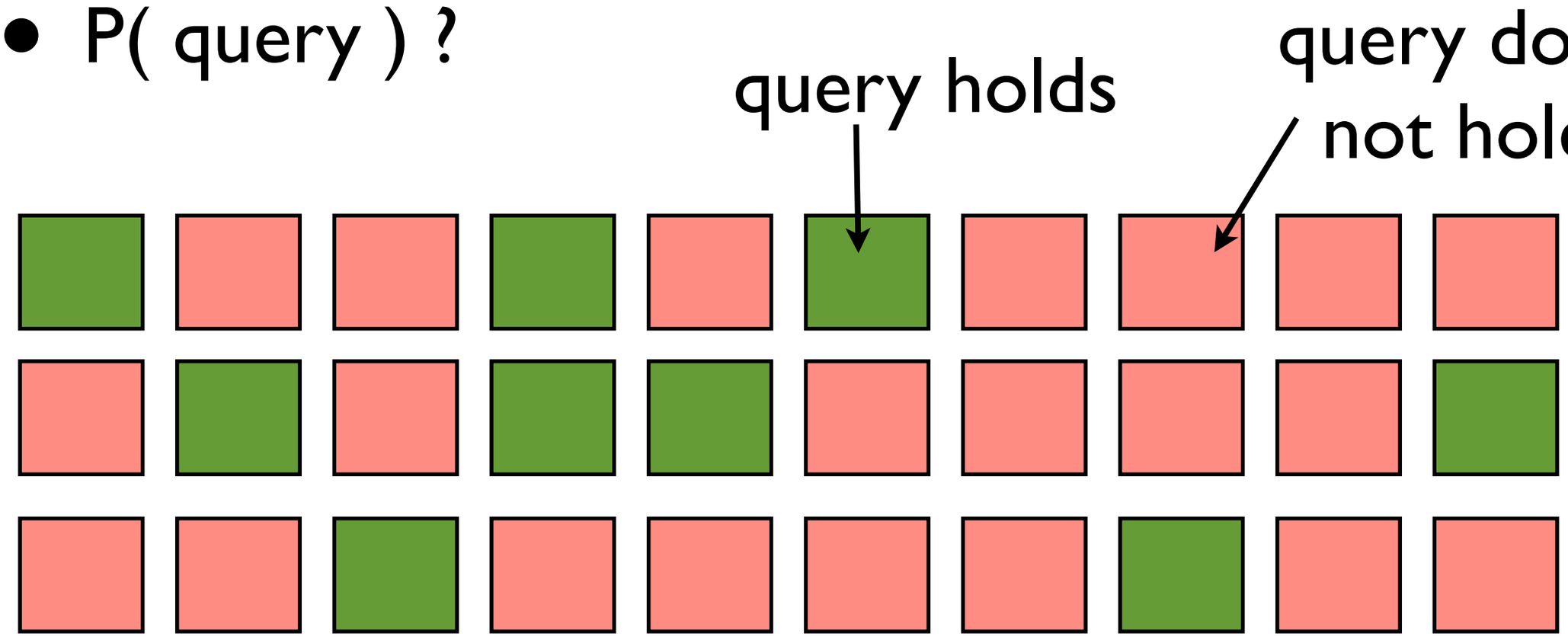
$$\phi_L \models \phi \models \phi_U$$

$$P(\phi_L) \leq P(\phi) \leq P(\phi_U)$$

- Sampling

Sampling

- P(query) ?



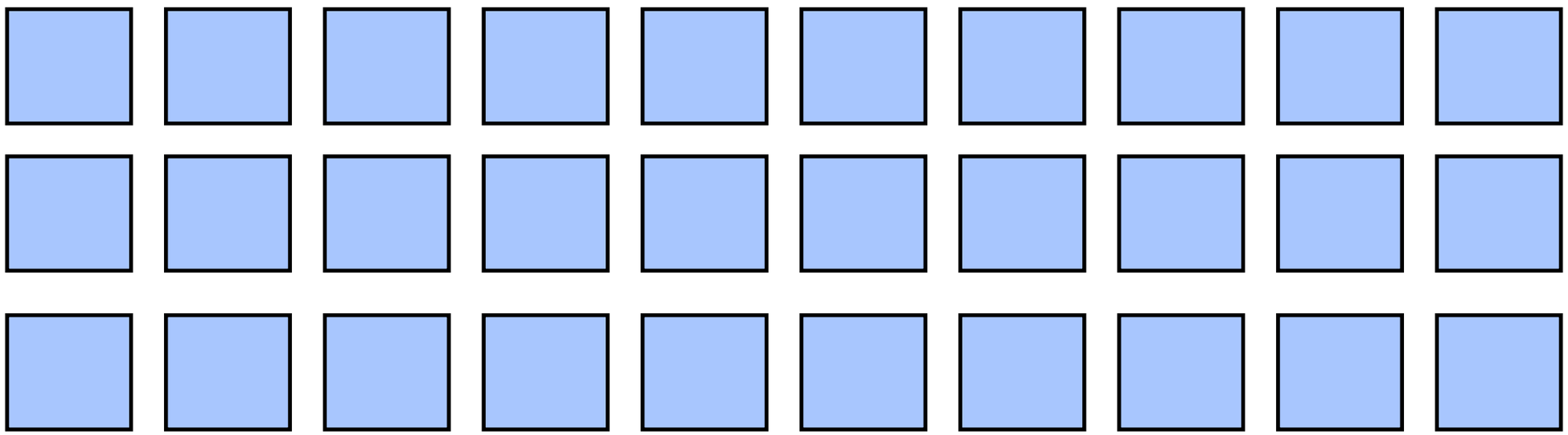
$$P(\text{query}) \approx \frac{\#\text{ query holds}}{\#\text{ worlds sampled}}$$

Rejection Sampling

- P(query | evidence) ?

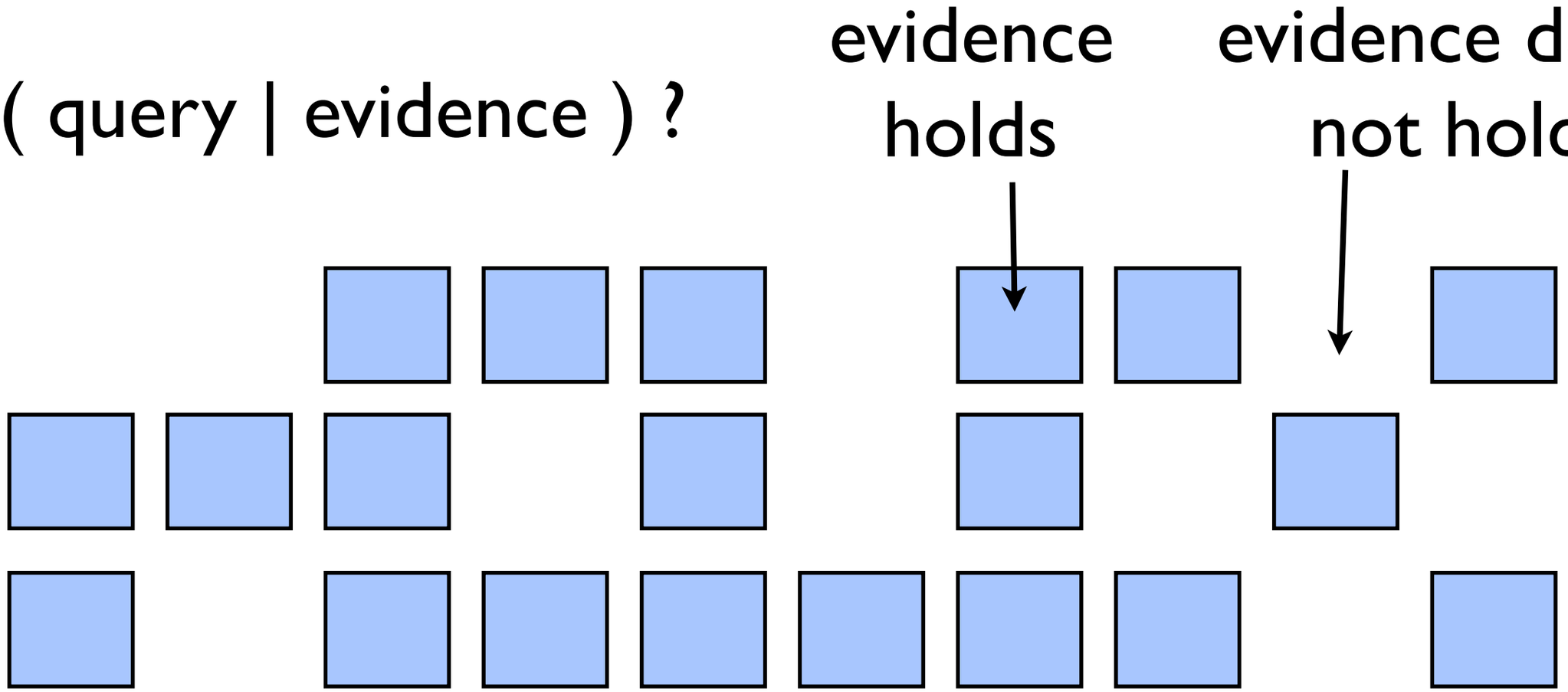
Rejection Sampling

- P(query | evidence) ?

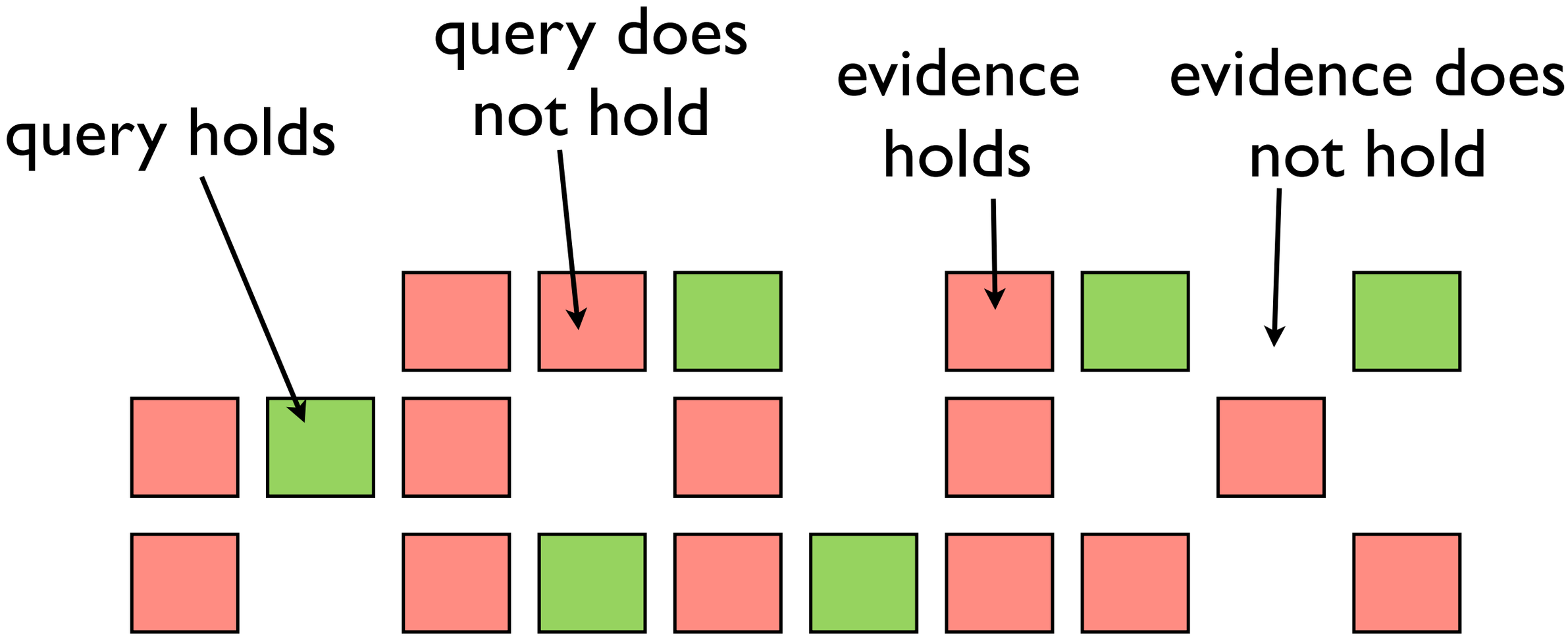


Rejection Sampling

- P(query | evidence) ?



Rejection Sampling



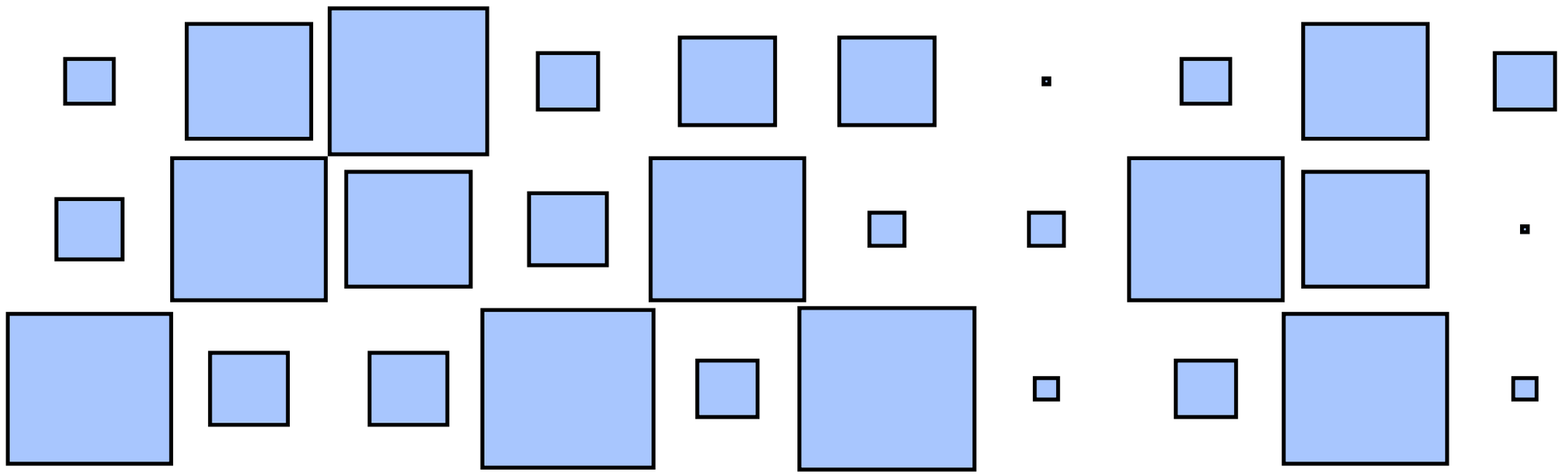
$$P(\text{query} \mid \text{evidence}) \approx \frac{\#\ \text{query \& evidence holds}}{\#\ \text{evidence holds}}$$

Likelihood Weighting

- P(query | evidence) ?

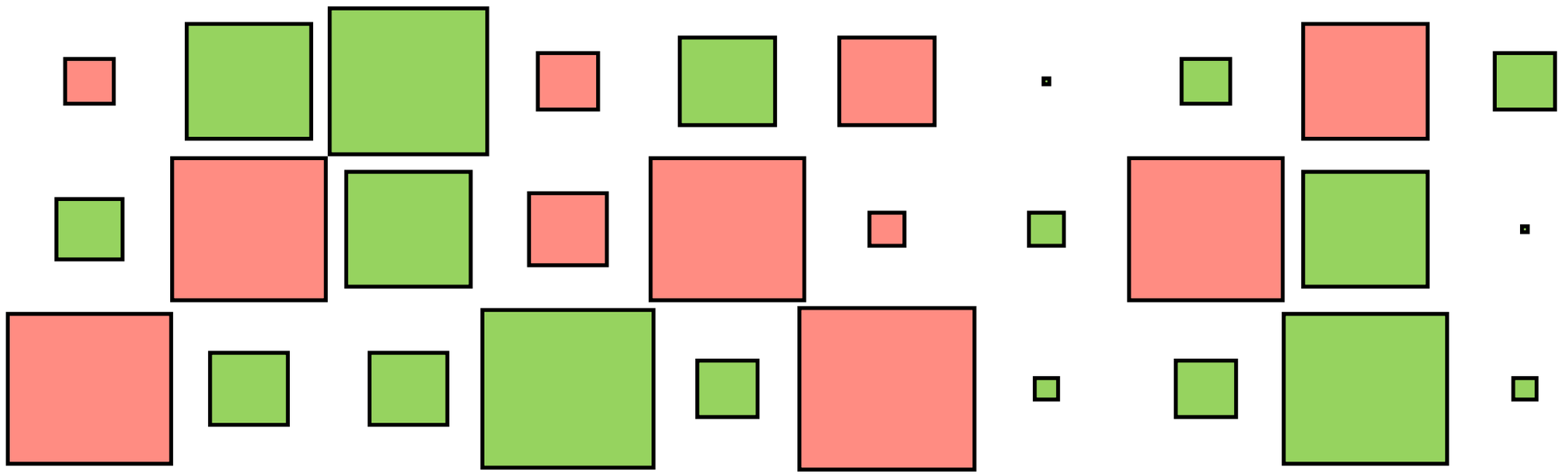
Likelihood Weighting

- P(query | evidence) ?



Likelihood Weighting

- P(query | evidence) ?



Markov Chain Monte Carlo (MCMC)

- Generate next sample by modifying current one
- Most common inference approach for PP languages such as Church, BLOG, ...
- Also considered for PRISM and ProbLog

Key challenges:
- how to propose next sample
- how to handle evidence

Roadmap

- Modeling
- Inference
- Learning & Dynamics
- Advanced Inference and KBMC

... with some detours on the way

Part III : Learning

Part III : Learning

a. Parameters

Parameter Learning

e.g., webpage classification model

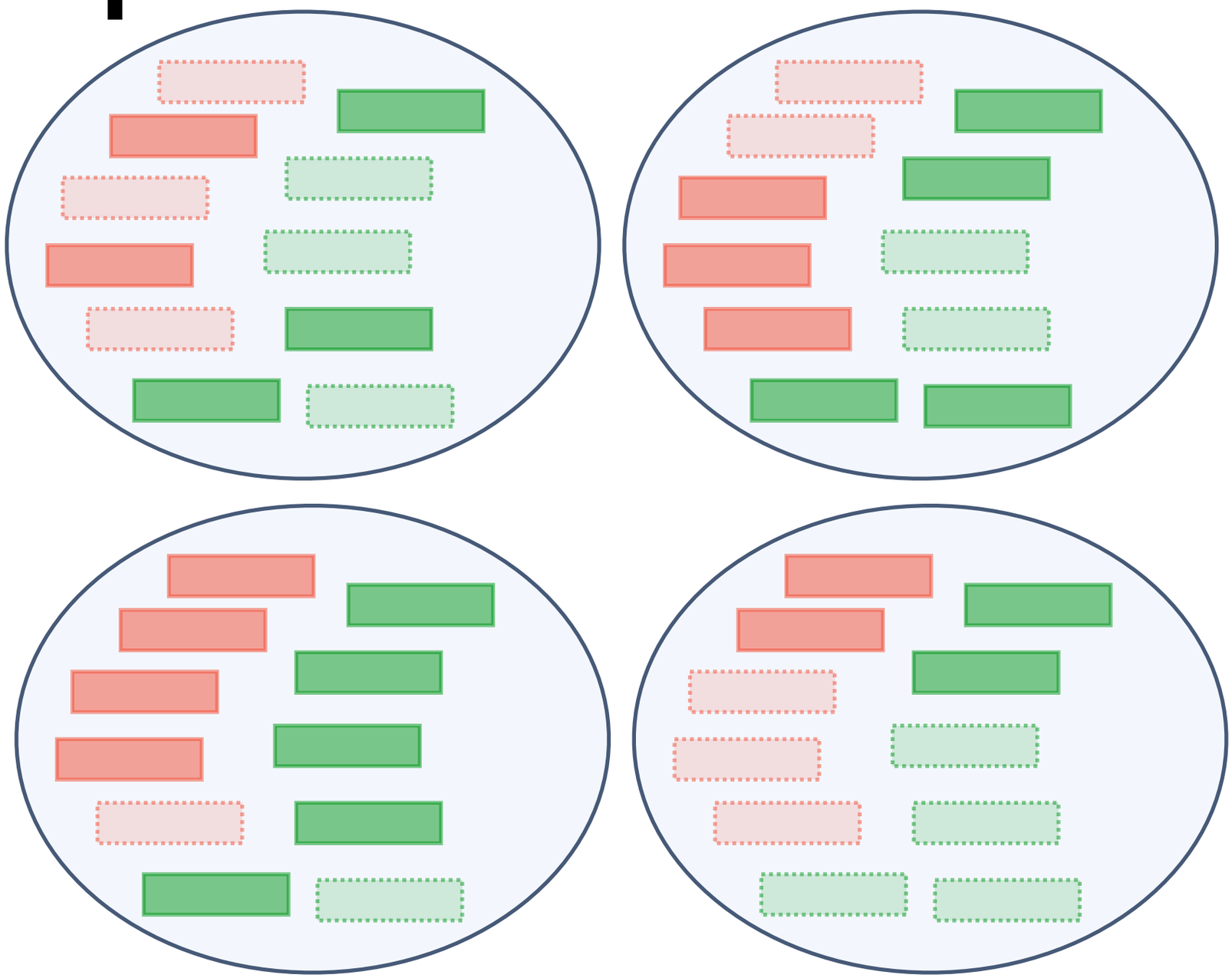
for each *CLASS1*, *CLASS2* and each *WORD*

?? :: link_class(Source,Target,*CLASS1*,*CLASS2*).
?? :: word_class(*WORD*,*CLASS*).

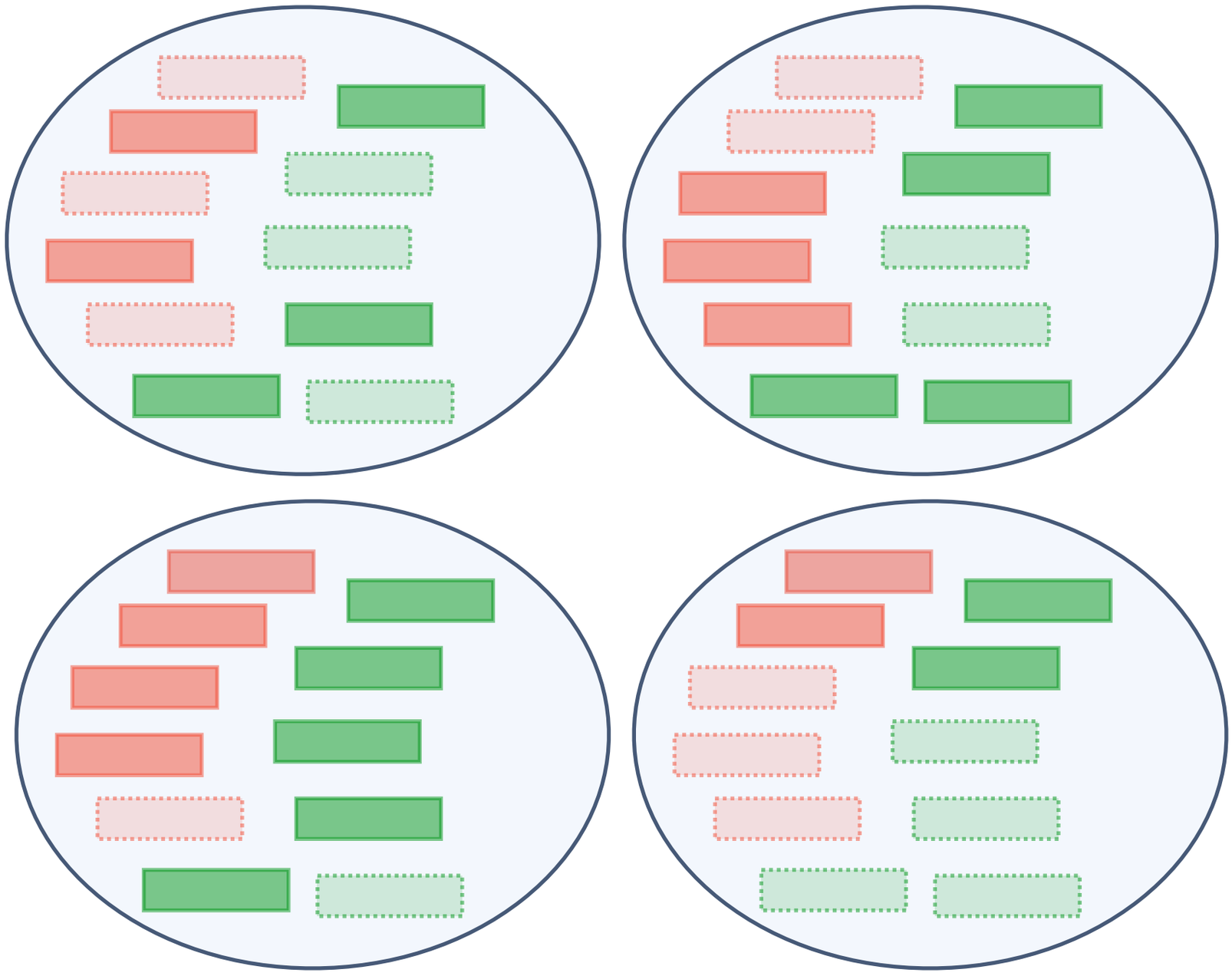
```
class(Page,C)    :-  has_word(Page,W), word_class(W,C).

class(Page,C)        :-  links_to(OtherPage,Page),
    class(OtherPage,OtherClass),
                         link_class(OtherPage,Page,OtherClass,C).
```

Sampling Interpretations

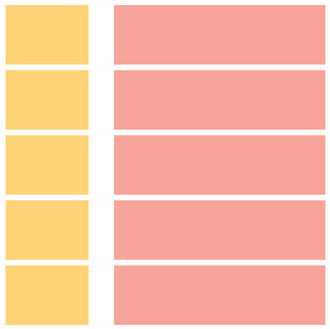
Sampling Interpretations

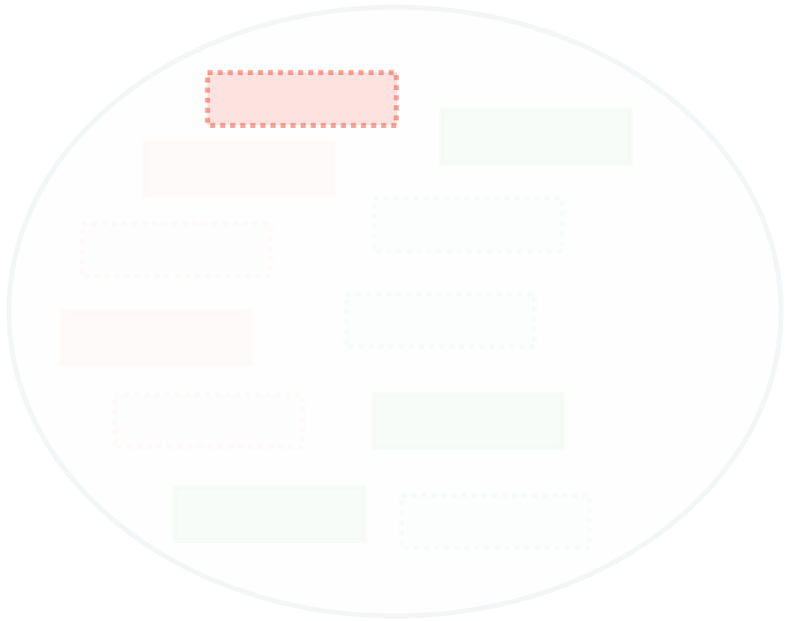


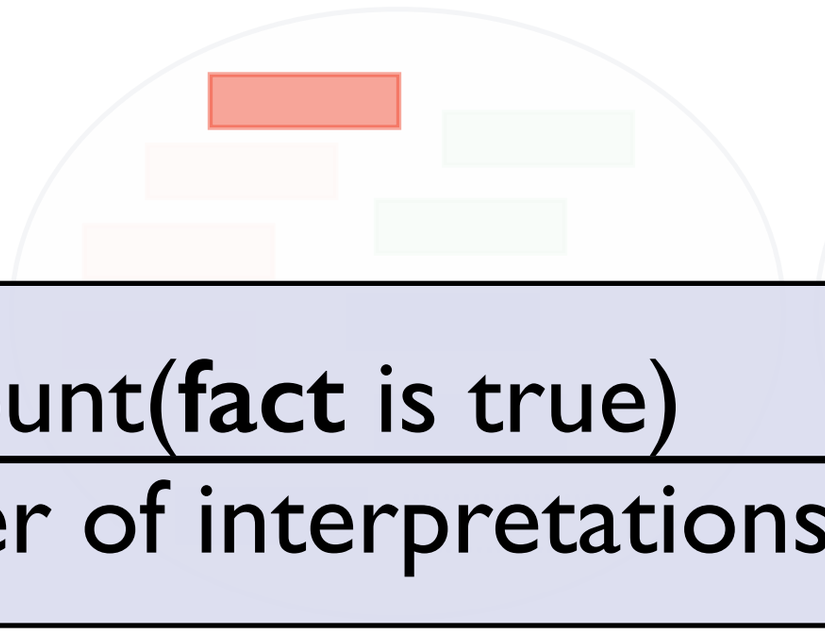
Parameter Estimation



Parameter Estimation

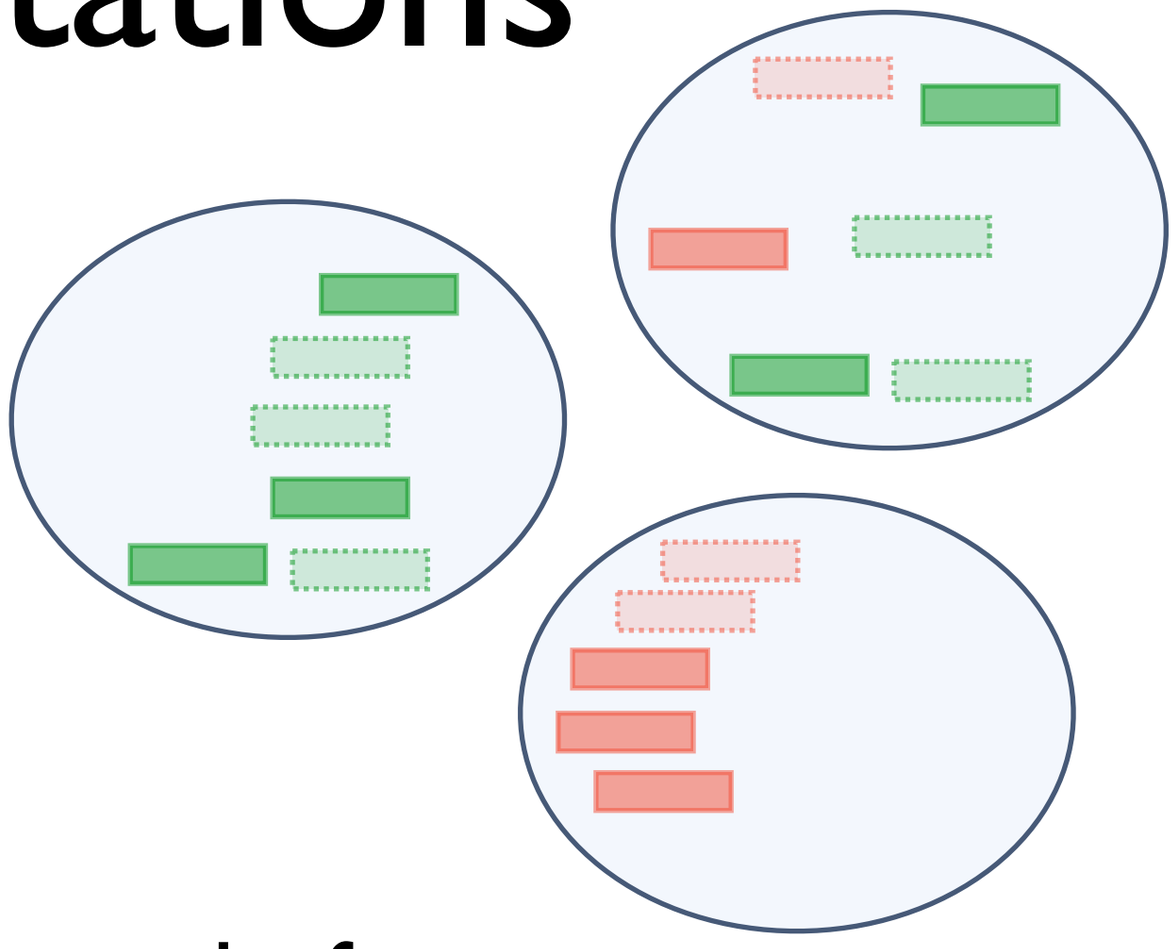






$$p(\textbf{fact}) = \frac{\text{count}(\textbf{fact} \text{ is true})}{\text{Number of interpretations}}$$

Learning from partial interpretations



- Not all facts observed
- Soft-EM
- use **expected count** instead of **count**
- **P(Q |E) -- conditional queries !**

 [Gutmann et al, ECML 11; Fierens et al, TPLP 14]

Learning from single facts / entailment

- Only true facts are given; e.g. as in HMM
  - key setting in PRISM, also in ProbLog
- EM-based, variations exist
- use **expected count** instead of **count**
- **P(Q |E) -- conditional queries !**

[Gutmann et al, ECML 08; Sato 95]

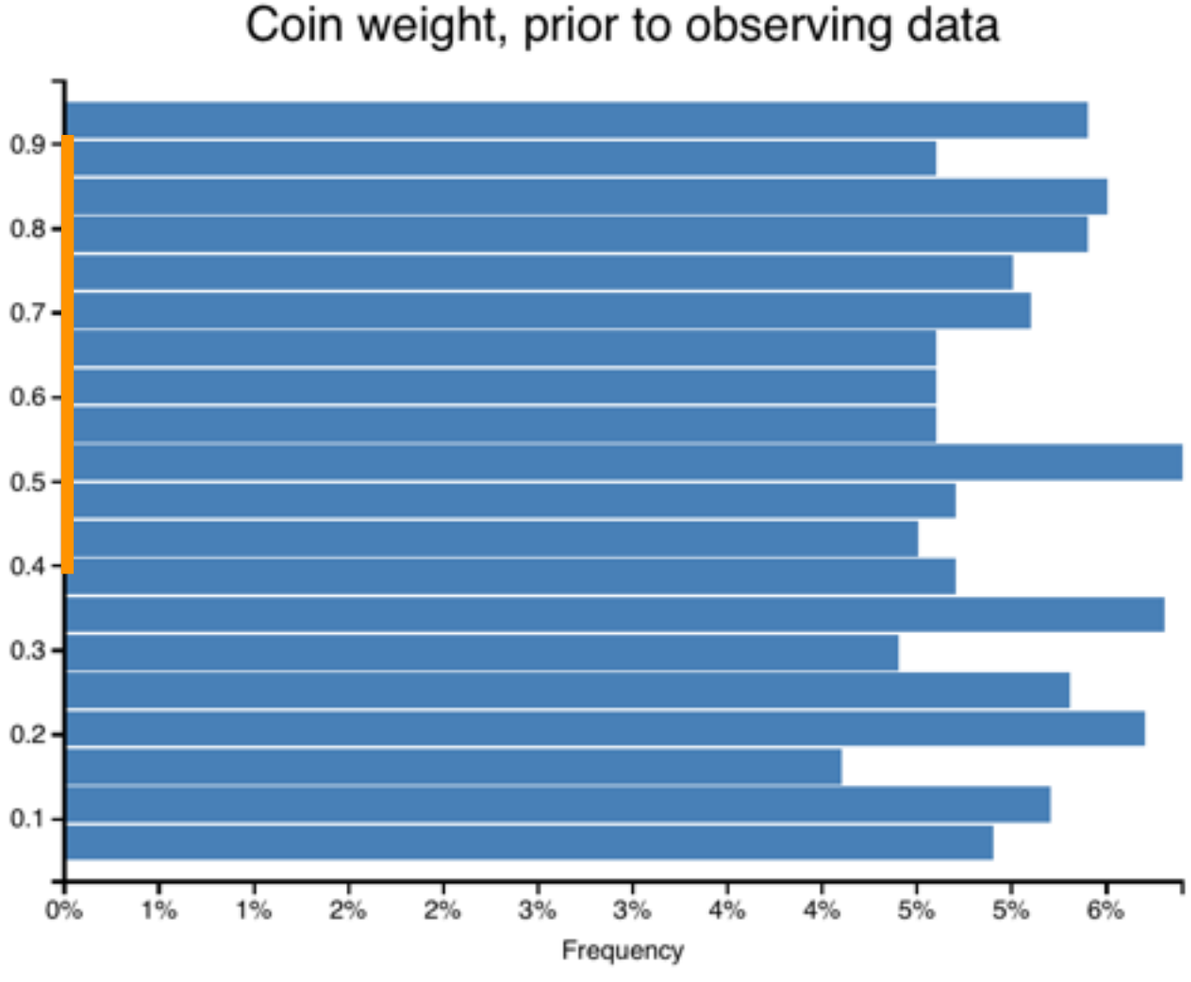
Bayesian Parameter Learning

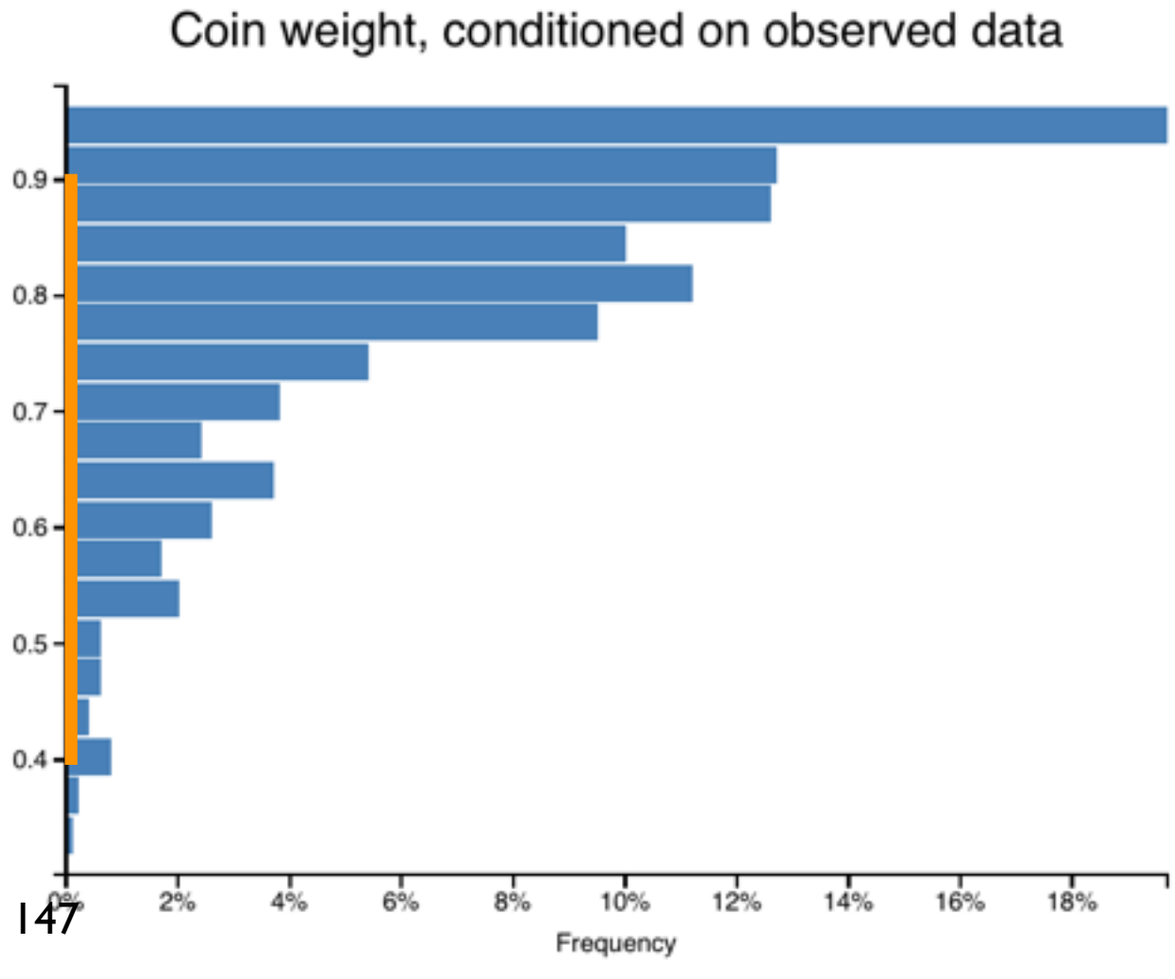
- Learning as inference (e.g., Church)
- Prior distributions for parameters
- Given data, find most likely parameter values

Example

[from probmods.org]

- Flipping a coin with unknown weight
- Prior: uniform distribution on [0,1]
- Observation: 5x heads in a row
- Sampling from Church model:





ProbLog Example

prior

```
0.05::weight(C,0.1); 0.2::weight(C,0.3); 0.5::weight(C,0.5);
                  0.2::weight(C,0.7); 0.05::weight(C,0.9) :- coin(C).
```

```
Param::toss(_,Param,_).
heads(C,R) :- weight(C,Param),toss(C,Param,R).
tails(C,R) :- weight(C,Param),\+toss(C,Param,R).

data(C,[]).
data(C,[h|R]) :- heads(C,R), data(C,R).
data(C,[t|R]) :- tails(C,R), data(C,R).
```

```
coin(c1).
coin(c2).
param(0.1).
param(0.3).
param(0.5).
param(0.7).
param(0.9).
```

```
query(weight(C,X)) :- coin(C),param(X).
```

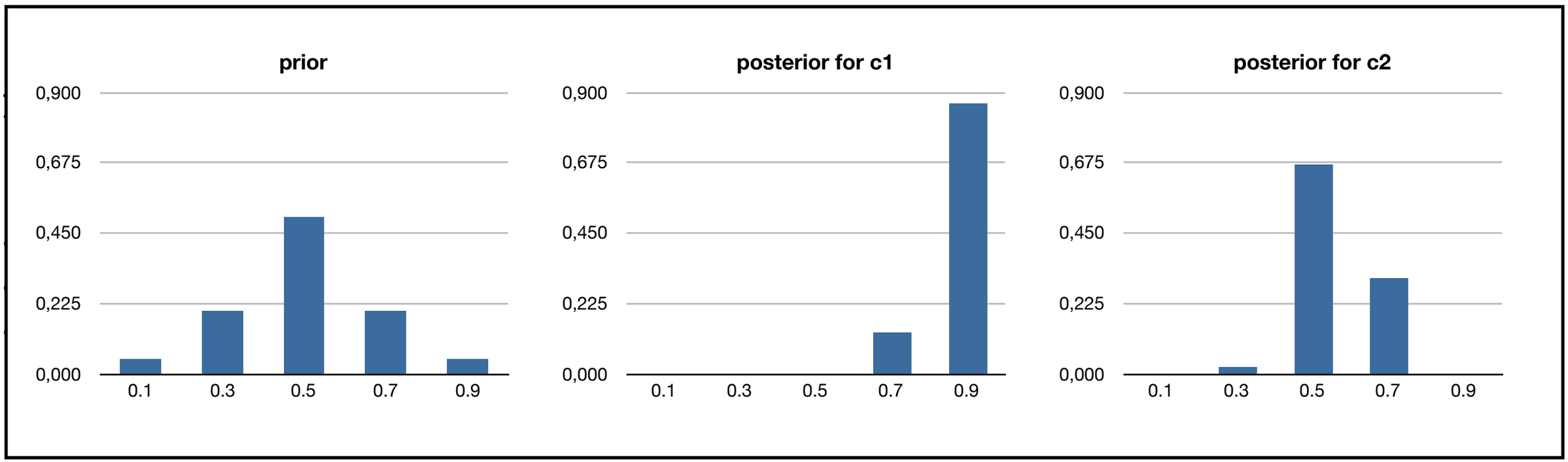
ask for posterior

```
evidence(data(c1,[h,h,h,h,h,h,h,h,h,h,h,h,h,h]),true).
evidence(data(c2,[h,t,h,h,h,h,h,t,t,h,t,t,h]),true).
```

data

ProbLog Example

```
0.05::weight(C,0.1); 0.2::weight(C,0.3); 0.5::weight(C,0.5);
                     0.2::weight(C,0.7); 0.05::weight(C,0.9) :- coin(C).
```




```
query(weight(C,X)) :- coin(C),param(X).

evidence(data(c1,[h,h,h,h,h,h,h,h,h,h,h,h,h,h]),true).
evidence(data(c2,[h,t,h,h,h,h,h,t,t,h,t,t,h]),true).
```

Part III : Learning

b. Rules / Structure

Information Extraction in NELL

Recently-Learned Facts twitter

Refresh

instance	iteration	date learned	confidence
kelly_andrews is a female	826	29-mar-2014	98.7
investment_next_year is an economic sector	829	10-apr-2014	95.3
shibenik is a geopolitical entity that is an organization	829	10-apr-2014	97.2
quality_web_design_work is a character trait	826	29-mar-2014	91.0
mercedes_benz_cls_by_carlsson is an automobile manufacturer	829	10-apr-2014	95.2
social_work is an academic program at the university rutgers_university	827	02-apr-2014	93.8
dante wrote the book the_divine_comedy	826	29-mar-2014	93.8
willie_aames was born in the city los_angeles	831	16-apr-2014	100.0
kitt_peak is a mountain in the state or province arizona	831	16-apr-2014	96.9
greenwich is a park in the city london	831	16-apr-2014	100.0

instances for many different relations

degree of certainty

NELL: http://rtw.ml.cmu.edu/rtw/

Rule learning in NELL

- Original approach
  - Make probabilistic data deterministic
  - run classic rule-learner (variant of FOIL)
  - re-introduce probabilities on learned rules and predict

ProbFOIL

- Upgrade rule-learning to a probabilistic setting within a relational learning / inductive logic programming setting
    - Works with a probabilistic logic program instead of a deterministic one.
- Introduce ProbFOIL, an adaption of Quinlan's FOIL to this setting.
- Apply to probabilistic databases like NELL

Pro Log

surfing(X) :- not pop(X) and windok(X).

H

surfing(X) :- not pop(X) and sunshine(X).

pop(e1). windok(e1). sunshine(e1). **B**

?-surfing(e1). **e**
no

B U H |=\= e (H does not cover e)

An ILP example

ProbLog

a probabilistic Prolog

p1:: surfing(X) :- not pop(X) and windok(X).

H

p2:: surfing(X) :- not pop(X) and sunshine(X).

0.2::pop(e1). 0.7::windok(e1). 0.6::sunshine(e1). **B**

?-P(surfing(e1)).**e**

gives (1-0.2) x 0.7 x p1 + (1-0.2) x 0.6 x (1-0.7) x p2 = P(B U H |= e)

not pop x windok x p1 + not pop x sunshine x (not windok) x p1

probability that the example is covered

Inductive Probabilistic Logic Programming

Given

a set of example facts e ∈ E together with the probability p that they hold

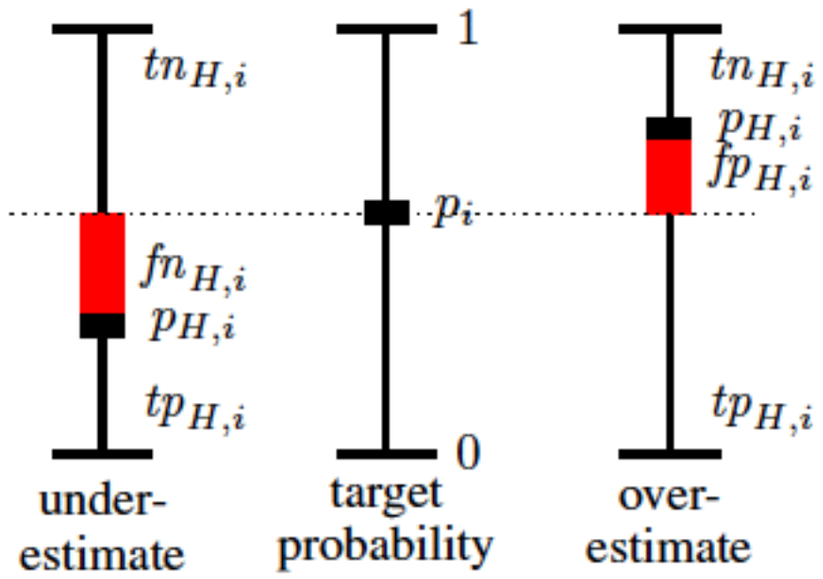
a background theory B in ProbLog

a hypothesis space L (a set of clauses)

Find

$$\arg\min_H loss(H, B, E) = \arg\min_H \sum_{e_i \in E} |P_s(B \cup H \models e) - p_i|$$

Adapt Rule-learner



Contingency table:
not only 1 / 0 values

Covering:
use multiple rules
to cover an example

Algorithm 1 The ProbFOIL$^+$ learning algorithm.

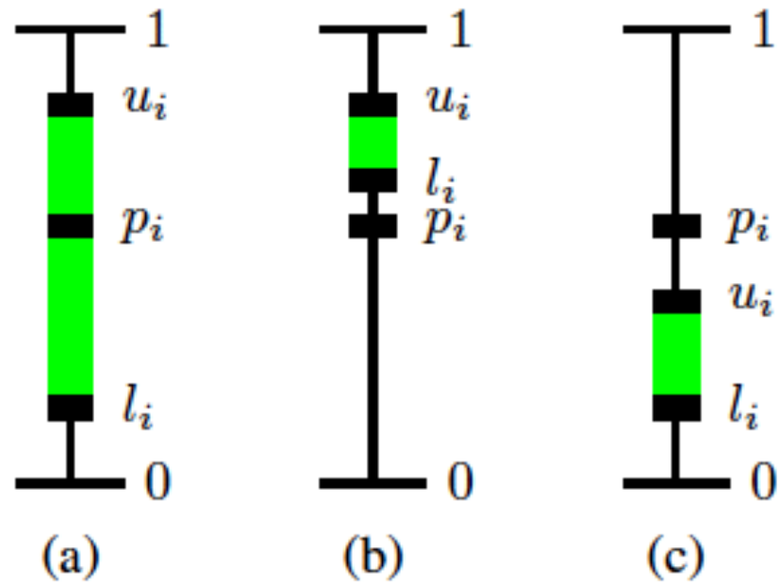
```
1:  function PROBFOIL+(target)
2:    H := ∅
3:    while true do
4:      clause := LEARNRULE(H, target)
5:      if GSCORE(H) < GSCORE(H ∪ {clause}) then
6:        H := H ∪ {clause}
7:      else return H
8:  function LEARNRULE(H, target)
9:    candidates := {x :: target ← true}
10:   best := (x :: target ← true)
11:   while candidates ≠ ∅ do
12:     next_cand := ∅
13:     for all x :: target ← body ∈ candidates do
14:       for all refinement ∈ ρ(target ← body) do
15:         if not REJECT(H, best, x :: target ← body) then
16:           next_cand := next_cand ∪ {x :: target ← body∧
17:                                     refinement}
18:           if LSCORE (H, x :: target ← body ∧ refinement) >
19:                                     LSCORE(H, best) then
20:             best := (x :: target ← body ∧ refinement)
21:     candidates := next_cand
22:   return best
```

Technical Novelty

p:: surfing(X) :- not pop(X) and windok(X).

u_i = (p=1)

l_i = (p=0)



ProbFOIL includes

a method to determine “optimal” p for a given rule

Experiments

Table 4: Precision for different experimental setups and parameters (A: $m = 1, p = 0.99$, B: $m = 1000, p = 0.90$).

Setting	**athleteplaysforteam**		**athleteplayssport**		**teamplaysinleague**		**athleteplaysinleague**		**teamplaysagainstteam**	
train/test/rule	**A**	**B**	**A**	**B**	**A**	**B**	**A**	**B**	**A**	**B**
1: det/det/det	74.00	69.36	94.14	93.47	96.29	82.15	80.95	74.14	73.40	73.86
2: det/prob/det	73.51	69.57	97.53	94.85	96.70	87.83	90.83	77.73	73.70	73.35
3: det/prob/prob	74.67	69.82	95.86	94.74	96.35	82.57	82.26	75.29	73.84	74.34
4: det/prob/prob	77.25	73.87	96.53	96.04	98.00	90.59	84.91	79.36	77.26	77.83
5: det/prob/prob	74.76	69.97	95.85	94.69	96.44	82.51	81.99	75.07	73.90	74.16
6: prob/prob/det	75.83	73.11	93.40	93.76	94.44	93.67	79.41	79.42	80.87	80.60
7: prob/prob/prob	78.31	73.72	95.62	95.10	98.84	91.86	96.94	79.49	85.78	81.81

Table 3: Learned relational rules for the different predicates (fold 1).

0.9375::athleteplaysforteam(A,B)	←	athleteledsportsteam(A,B).
0.9675::athleteplaysforteam(A,B)	←	athleteledsportsteam(A,V1), teamplaysagainstteam(B,V1).
0.9375::athleteplaysforteam(A,B)	←	athleteplayssport(A,V1), teamplayssport(B,V1).
0.5109::athleteplaysforteam(A,B)	←	athleteplaysinleague(A,V1), teamplaysinleague(B,V1).
0.9070::athleteplayssport(A,B)	←	athleteledsportsteam(A,V2), teamalsoknownas(V2,V1), teamplayssport(V1,B), teamplayssport(V2,B).
0.9070::athleteplayssport(A,B)	←	athleteplaysforteam(A,V2), teamalsoknownas(V2,V1), teamplayssport(V1,B), teamplayssport(V2,B),teamalsoknownas(V1,V2).
0.9070::athleteplayssport(A,B)	←	athleteplaysforteam(A,V1), teamplayssport(V1,B).
0.9286::athleteplaysinleague(A,B)	←	athleteledsportsteam(A,V1), teamplaysinleague(V1,B).
0.7868::athleteplaysinleague(A,B)	←	athleteplaysforteam(A,V2), teamalsoknownas(V2,V1), teamplaysinleague(V1,B).
0.9384::athleteplaysinleague(A,B)	←	athleteplayssport(A,V2), athleteplayssport(V1,V2), teamplaysinleague(V1,B).
0.9024::athleteplaysinleague(A,B)	←	athleteplaysforteam(A,V1), teamplaysinleague(V1,B).

ProbFOIL

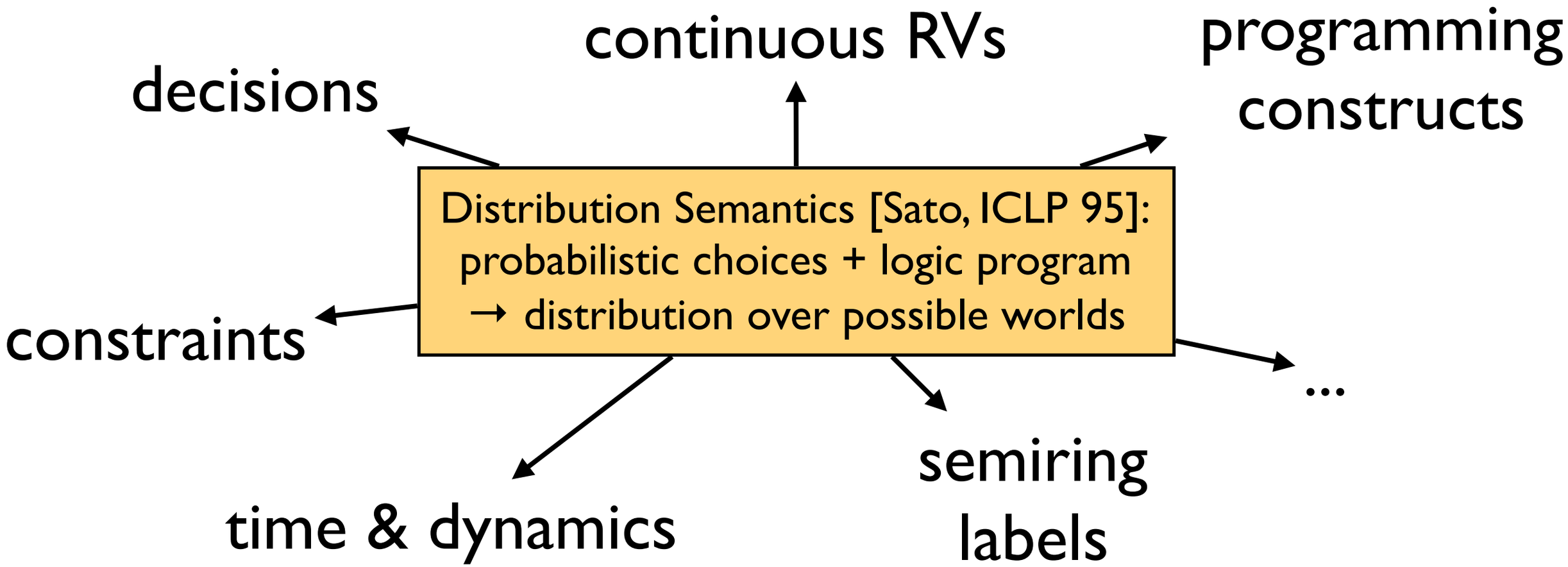
- Upgrade rule-learning to a probabilistic setting within a relational learning / inductive logic programming setting
    - Works with a probabilistic logic program instead of a deterministic one.
- Introduce ProbFOIL, an adaption of Quinlan's FOIL to this setting.
- Apply to probabilistic databases like NELL

Roadmap

- Modeling
- Inference
- Learning & Dynamics
- Advanced Inference and KBMC

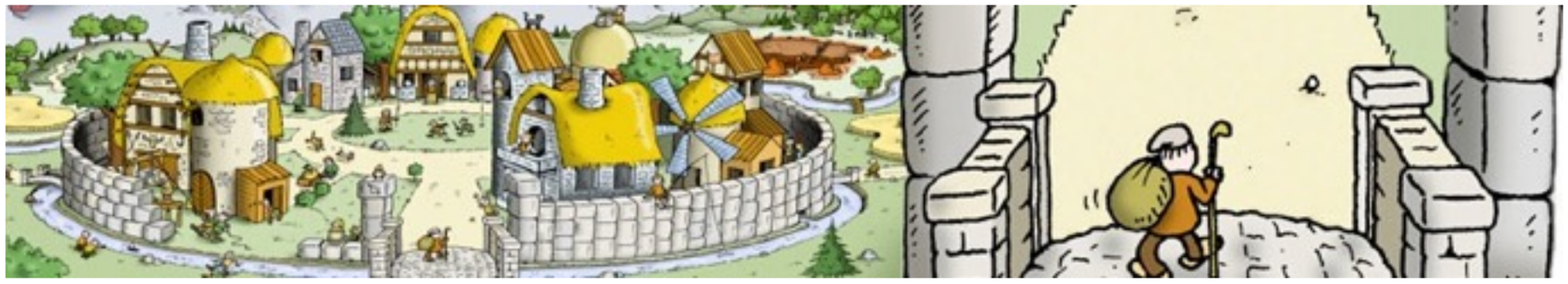
... with some detours on the way

Extensions of basic PLP



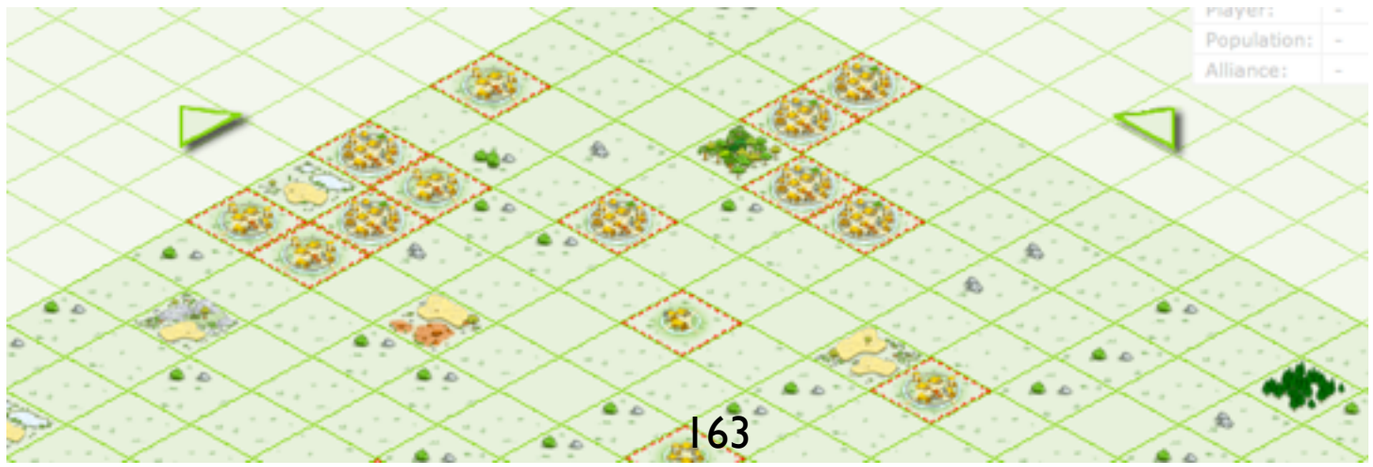
Part IV : Dynamics

Dynamics: Evolving Networks



- *Travian*: A massively multiplayer real-time strategy game
- Commercial game run by TravianGames GmbH
- ~3.000.000 players spread over different "worlds"
- ~25.000 players in one world

[Thon et al. ECML 08]



World Dynamics

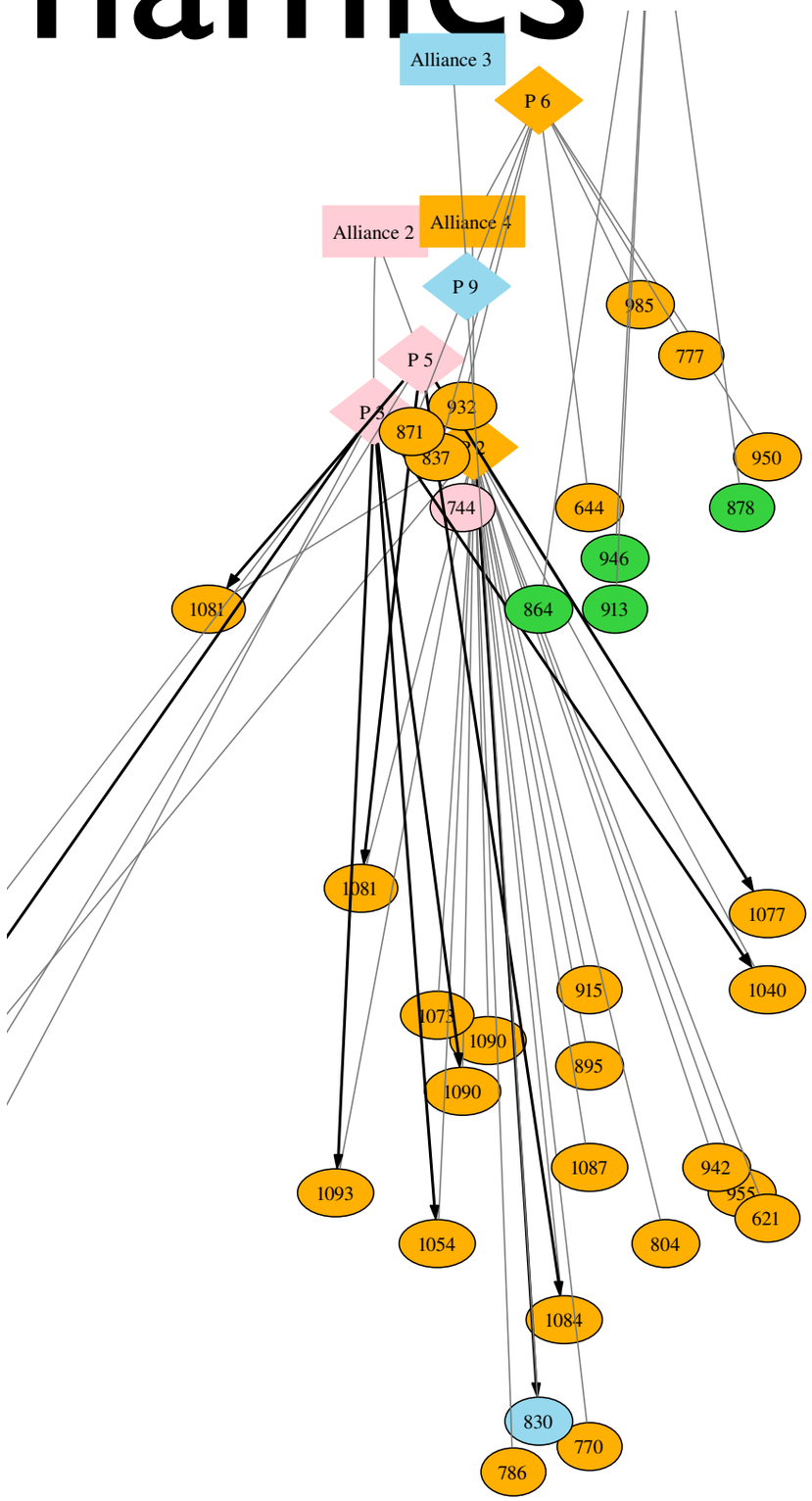
Fragment of world with

~10 alliances
~200 players
~600 cities

alliances color-coded

Can we build a model
of this world ?
Can we use it for playing
better ?

[Thon, Landwehr, De Raedt, ECML08]



World Dynamics

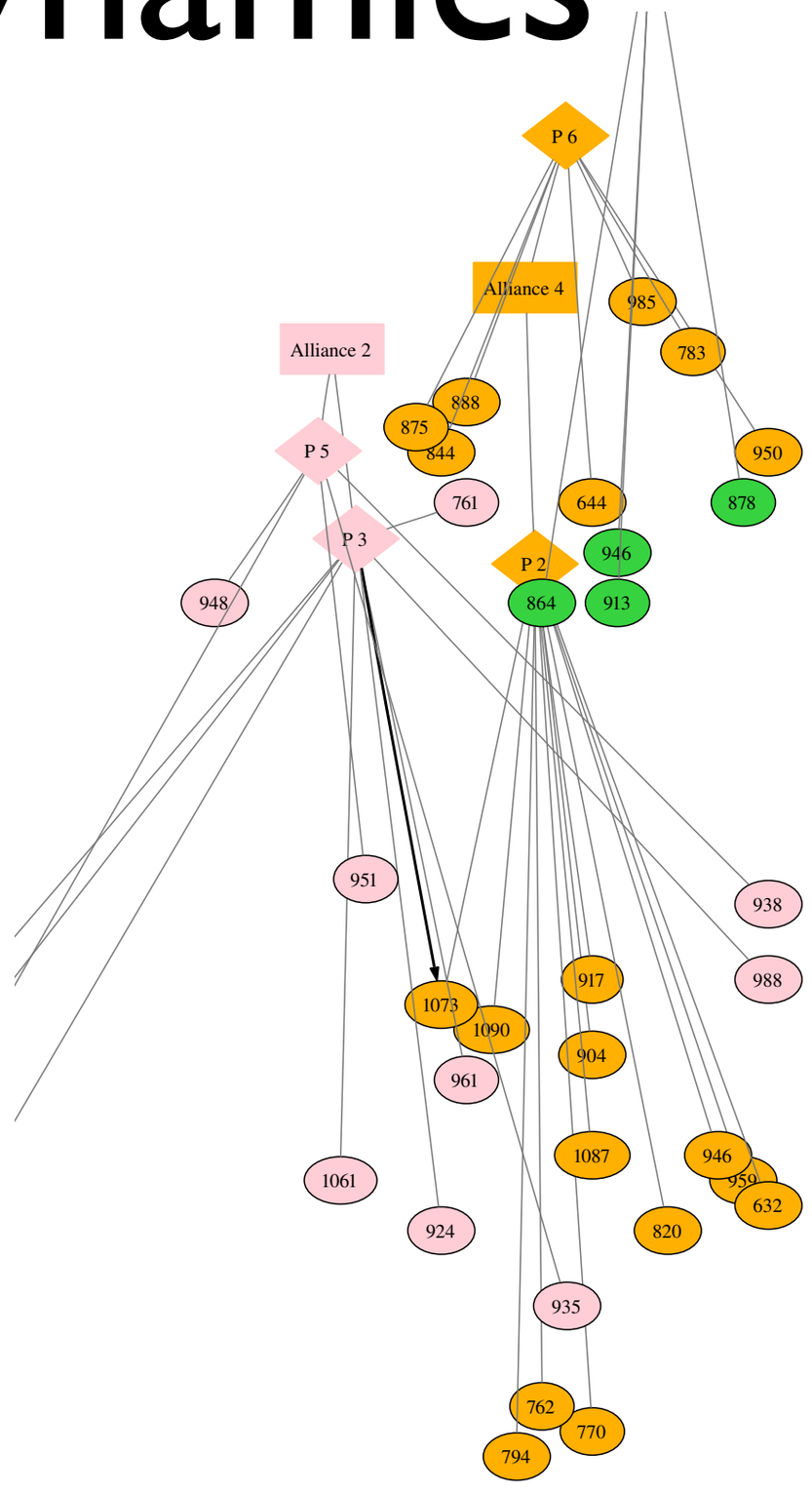
Fragment of world with

~10 alliances
~200 players
~600 cities

alliances color-coded

Can we build a model
of this world ?
Can we use it for playing
better ?

[Thon, Landwehr, De Raedt, ECML08]



World Dynamics

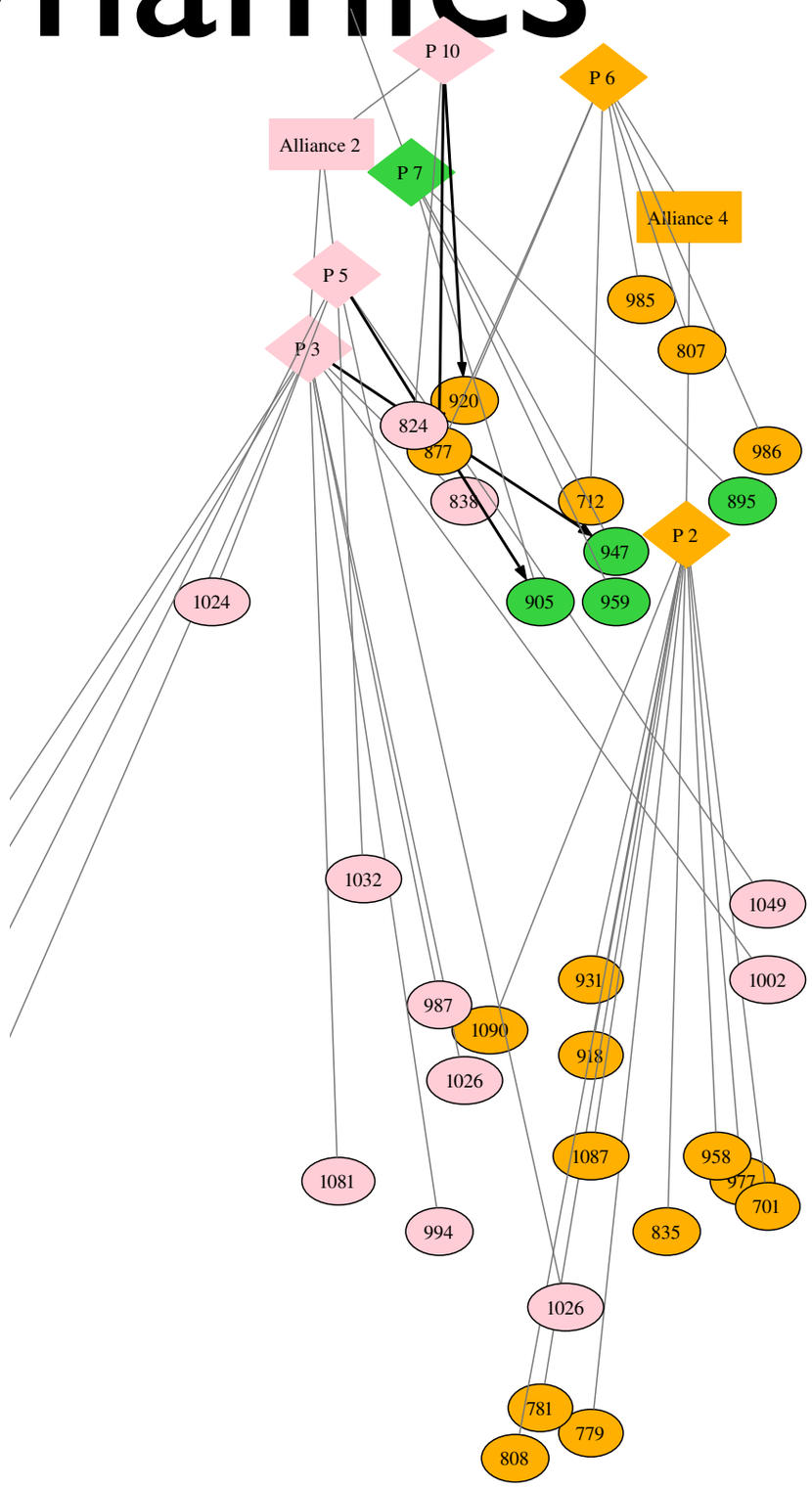
Fragment of world with

~10 alliances
~200 players
~600 cities

alliances color-coded

Can we build a model
of this world ?
Can we use it for playing
better ?

[Thon, Landwehr, De Raedt, ECML08]



World Dynamics

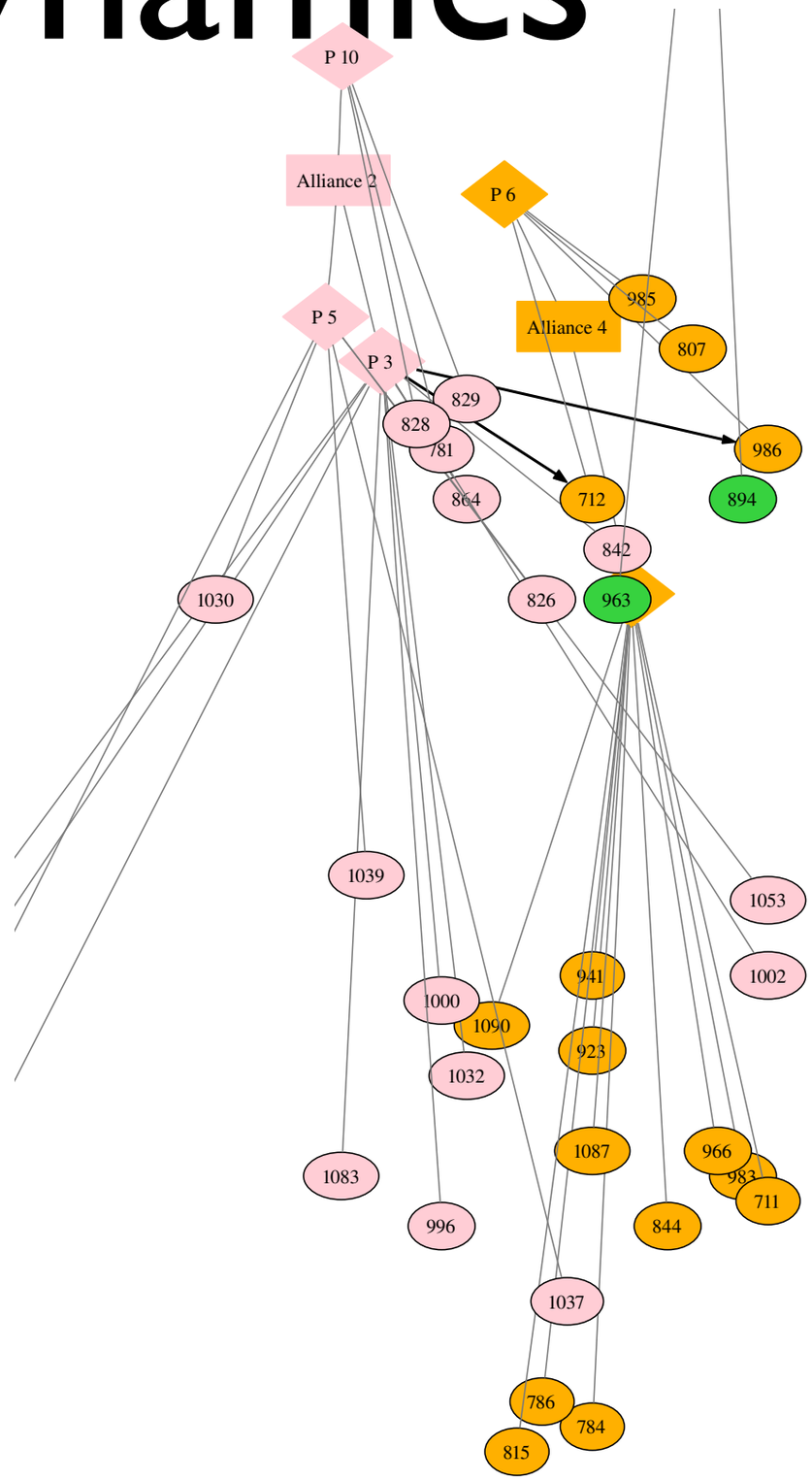
Fragment of world with

~10 alliances
~200 players
~600 cities

alliances color-coded

Can we build a model
of this world ?
Can we use it for playing
better ?

[Thon, Landwehr, De Raedt, ECML08]



World Dynamics

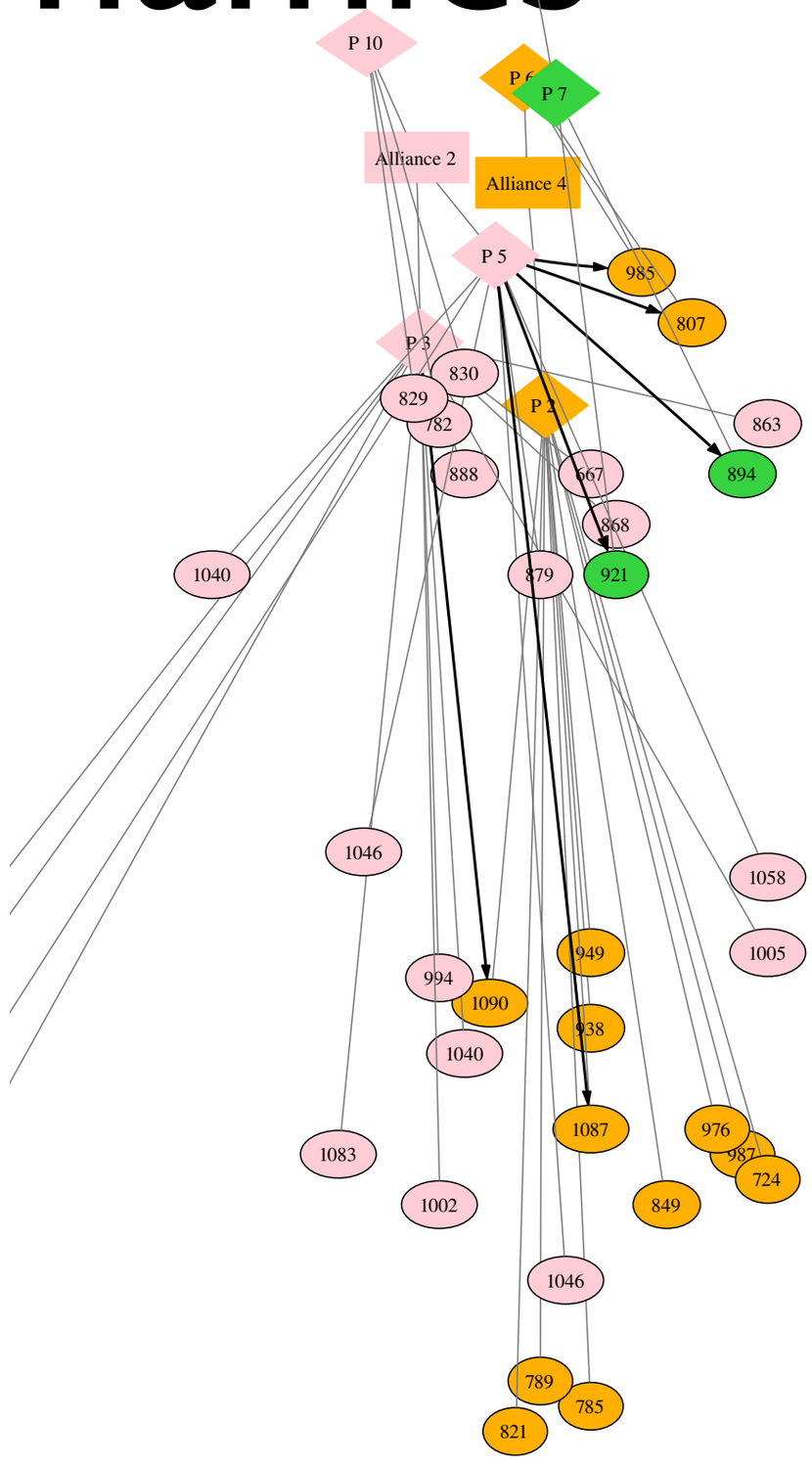
Fragment of world with

~10 alliances
~200 players
~600 cities

alliances color-coded

Can we build a model
of this world ?
Can we use it for playing
better ?

[Thon, Landwehr, De Raedt, ECML08]



World Dynamics

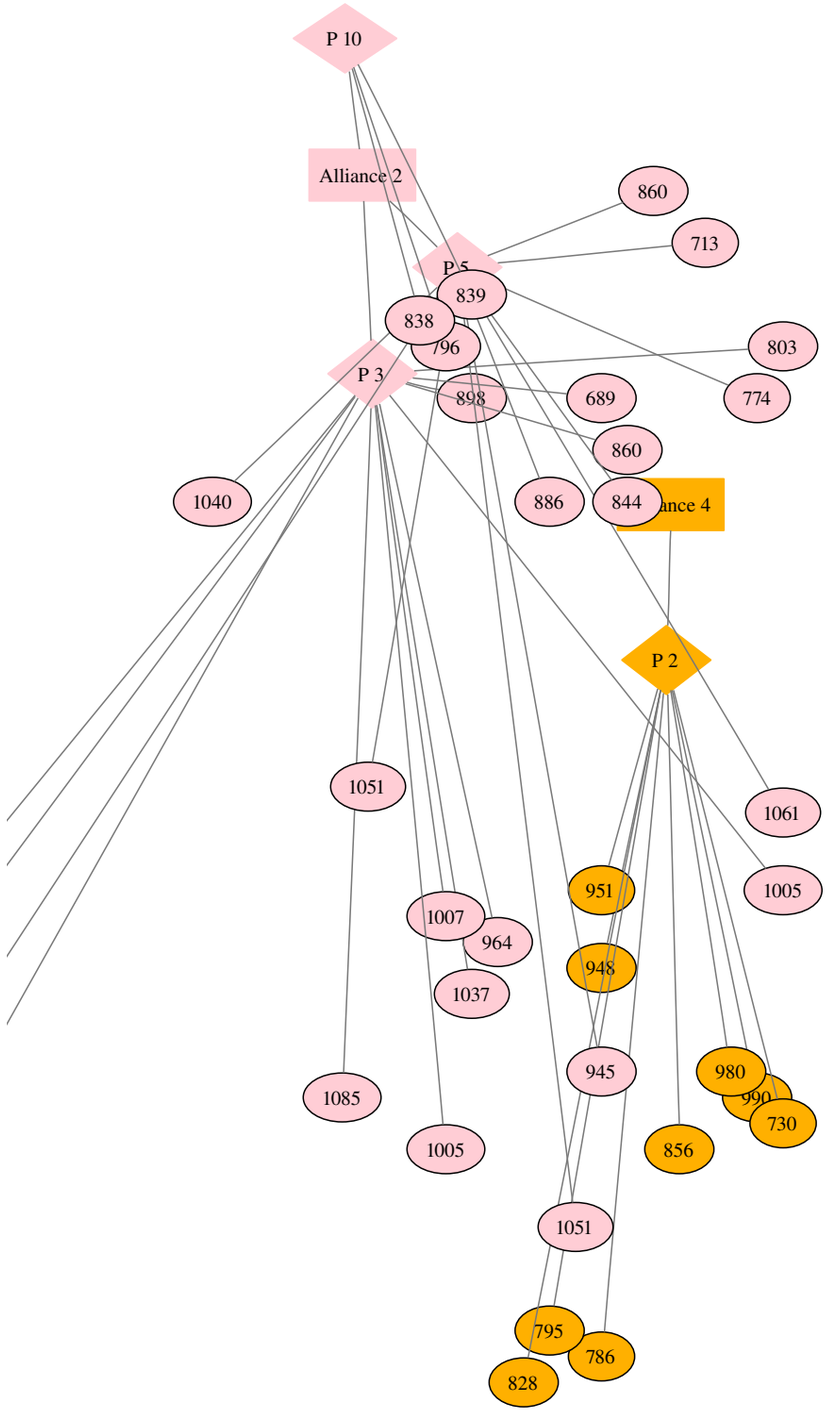
Fragment of world with

~10 alliances
~200 players
~600 cities

alliances color-coded

Can we build a model
of this world ?
Can we use it for playing
better ?

[Thon, Landwehr, De Raedt, ECML08]

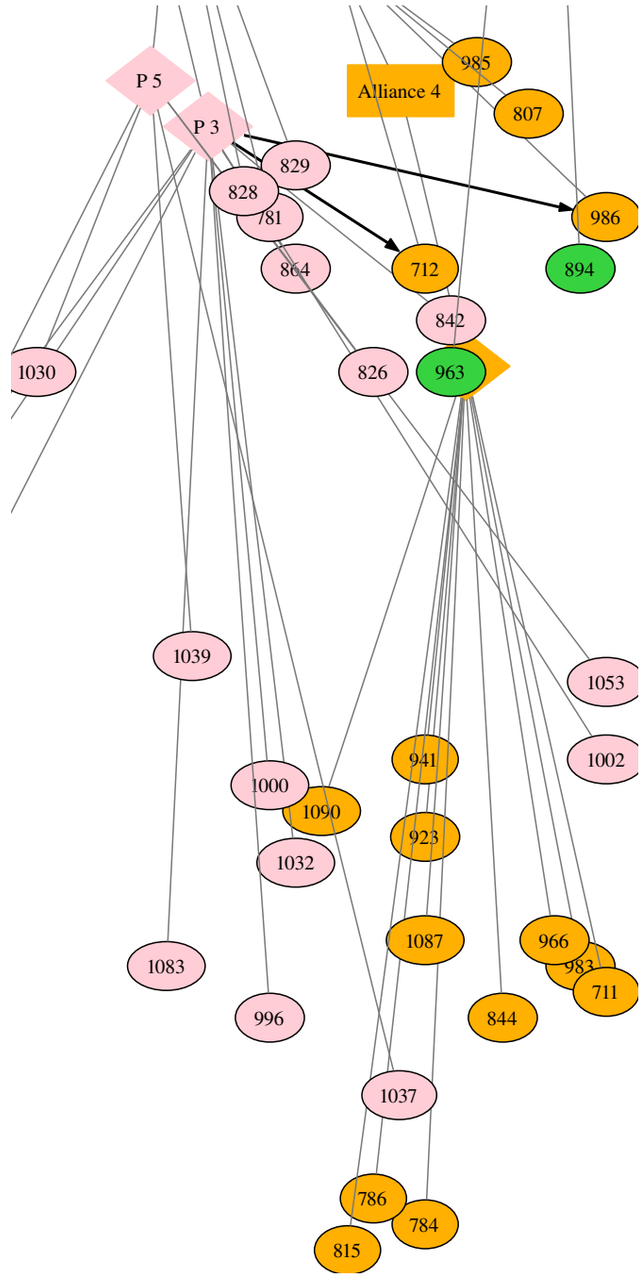


CPT-Rules

$$\frac{b_1, \ldots b_n}{\text{cause}} \rightarrow \frac{h_1 : p_1 \vee \ldots \vee h_m : p_m}{\text{effect}}$$

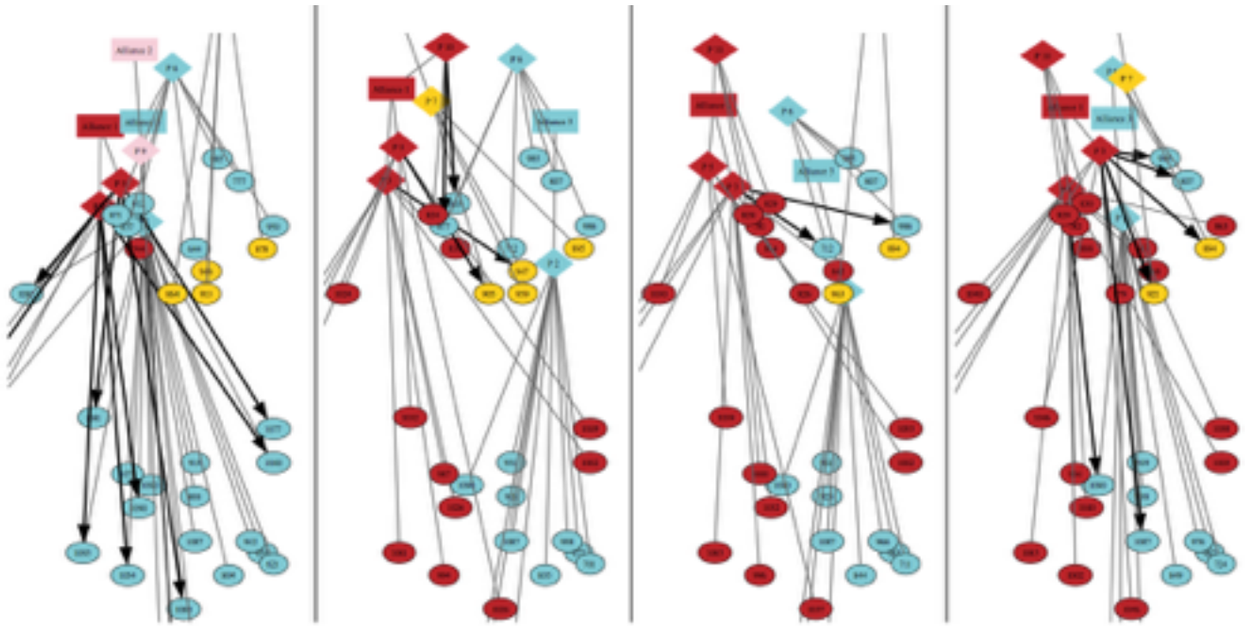
$$city(C, Owner), city(C2, Attacker), close(C, C2) \rightarrow$$
$$conquest(Attacker, C2) \ : \ p \ \vee \ nil \ : \ (1-p)$$

conquer a city which is close
P(conquest(),Time+5) ?
learn parameters



Thon et al. MLJ 11

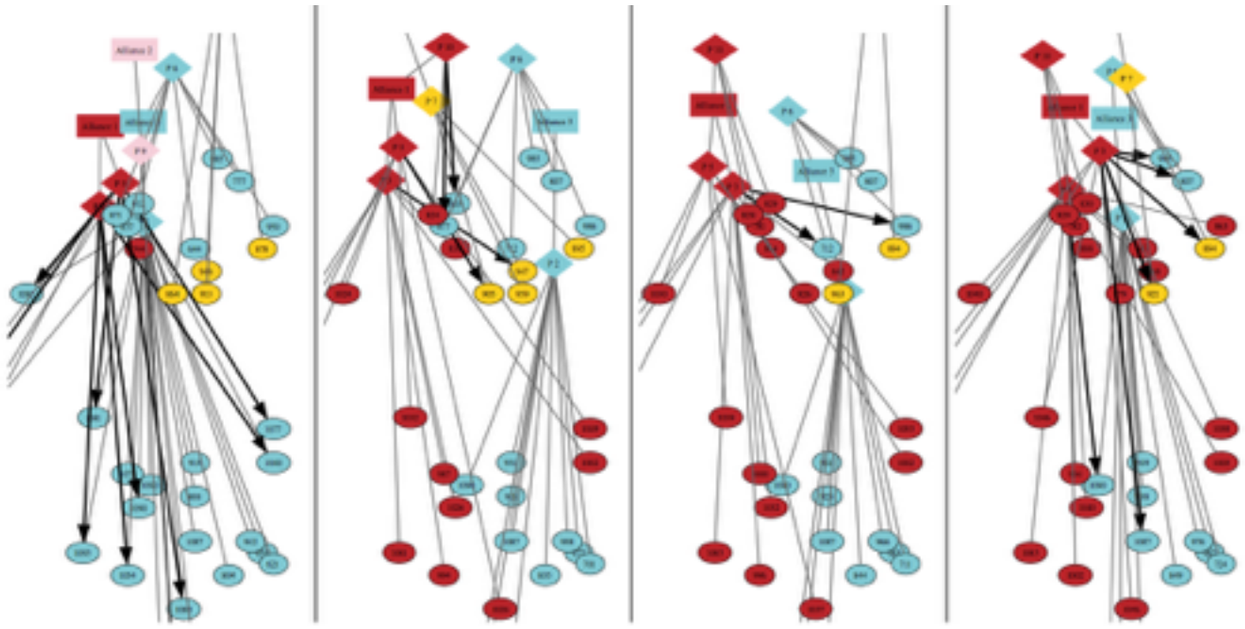
Causal Probabilistic Time-Logic (CPT-L)



how does the world change over time?

[Thon et al, MLJ 11]

Causal Probabilistic Time-Logic (CPT-L)



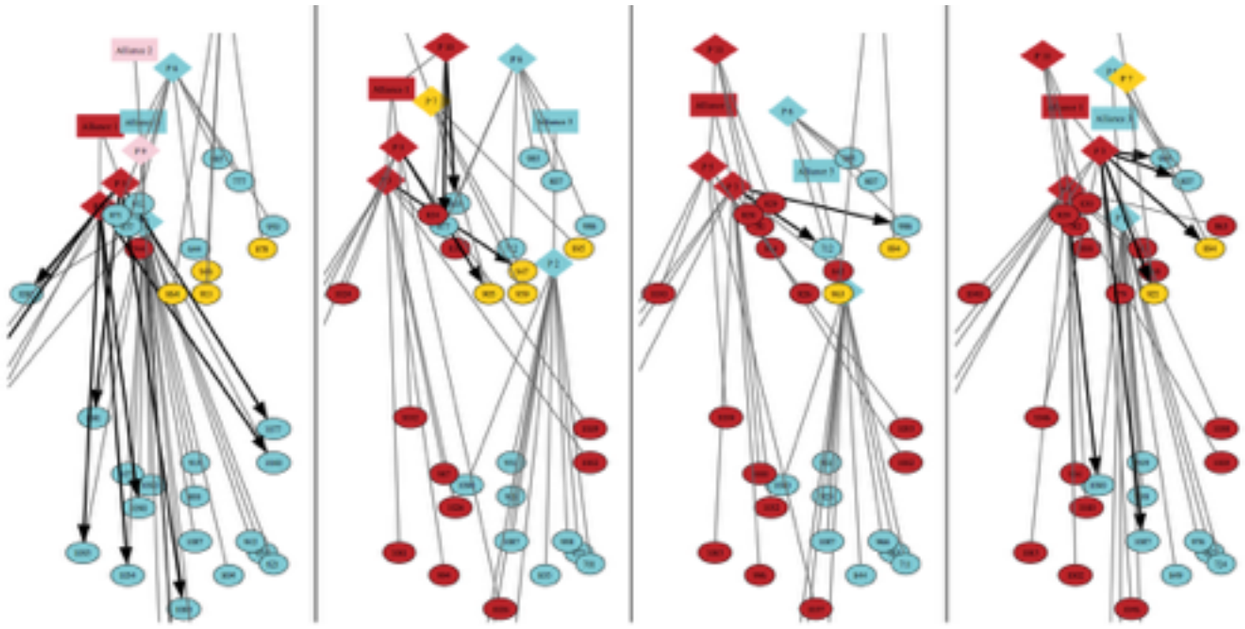
how does the world change over time?

```
0.4::conquest(Attacker,C); 0.6::nil :-
    city(C,Owner),city(C2,Attacker),close(C,C2).
```

if **cause** holds at time T

[Thon et al, MLJ 11]

Causal Probabilistic Time-Logic (CPT-L)



how does the world change over time?

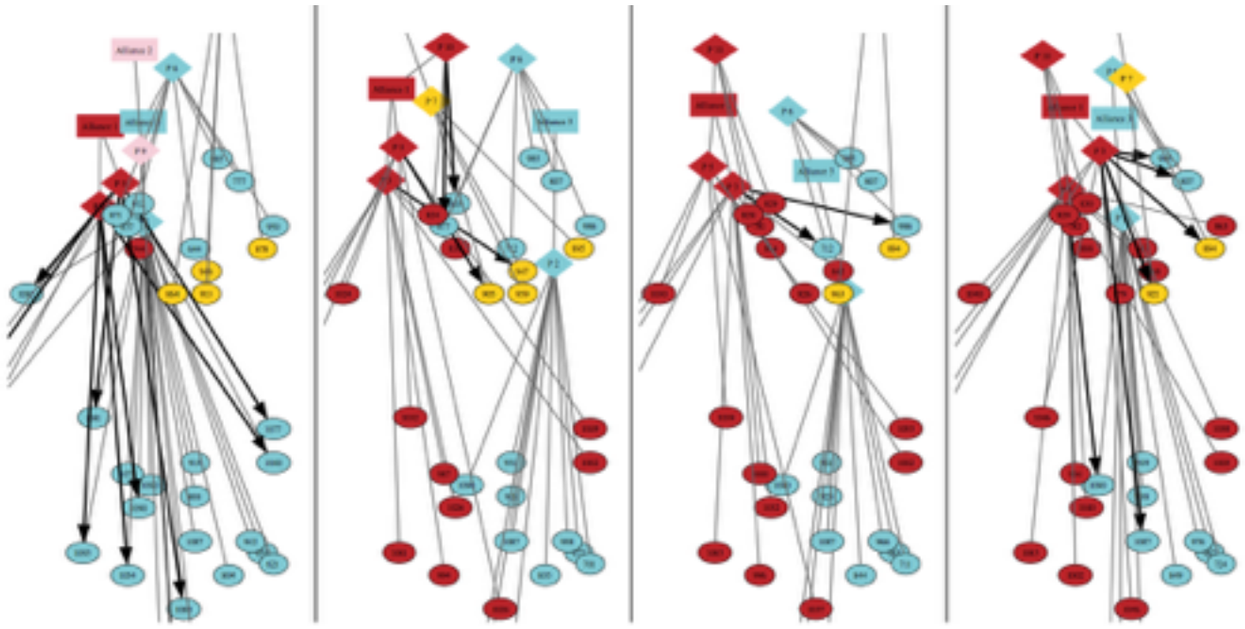
one of the **effects** holds at time T+1

```
0.4::conquest(Attacker,C); 0.6::nil :-
        city(C,Owner),city(C2,Attacker),close(C,C2).
```

if **cause** holds at time T

[Thon et al, MLJ 11]

Causal Probabilistic Time-Logic (CPT-L)



how does the world change over time?

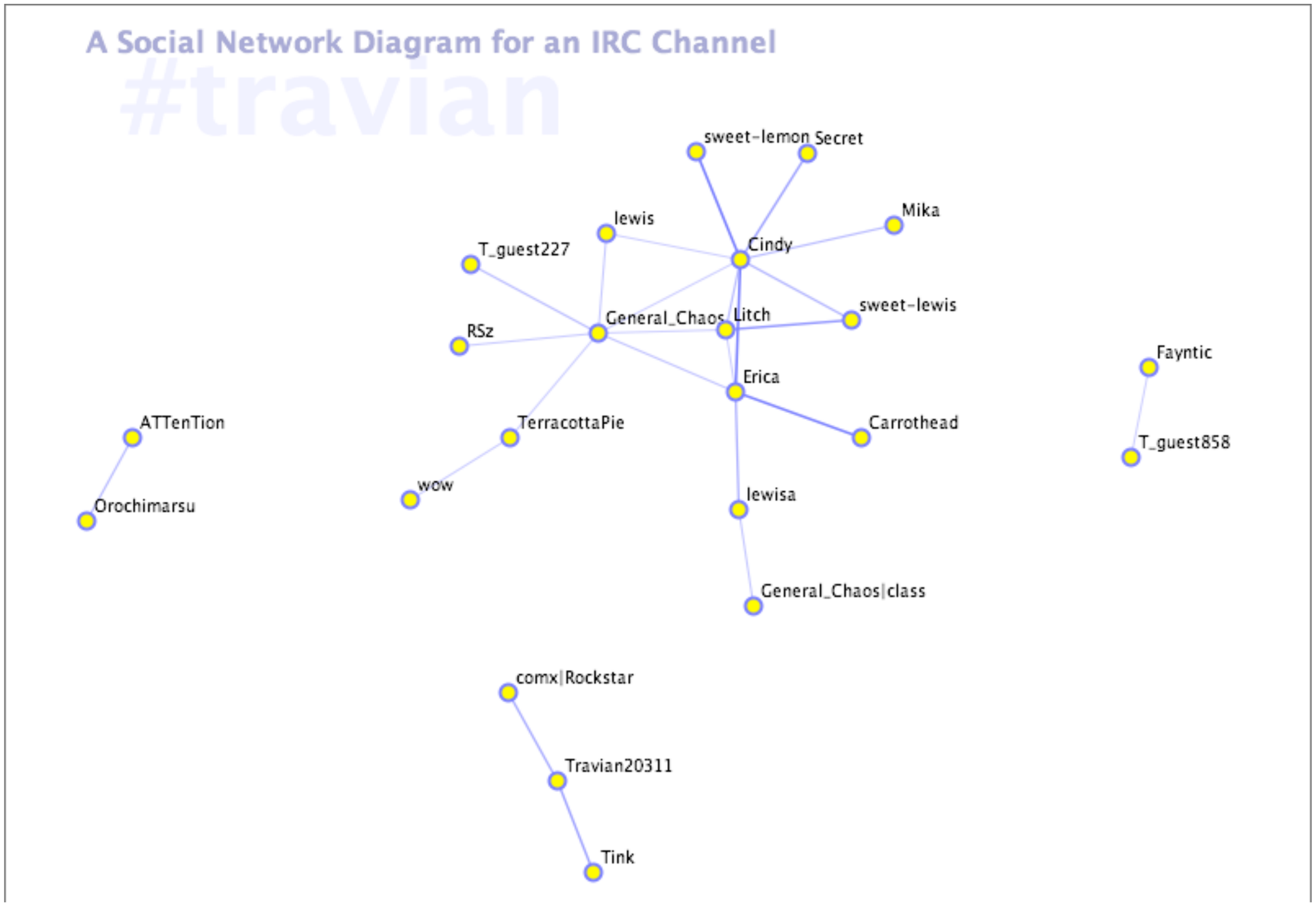
one of the **effects** holds at time T+1

```
0.4::conquest(Attacker,C); 0.6::nil :-
                    city(C,Owner),city(C2,Attacker),close(C,C2).
```

if **cause** holds at time T

[Thon et al, MLJ 11]

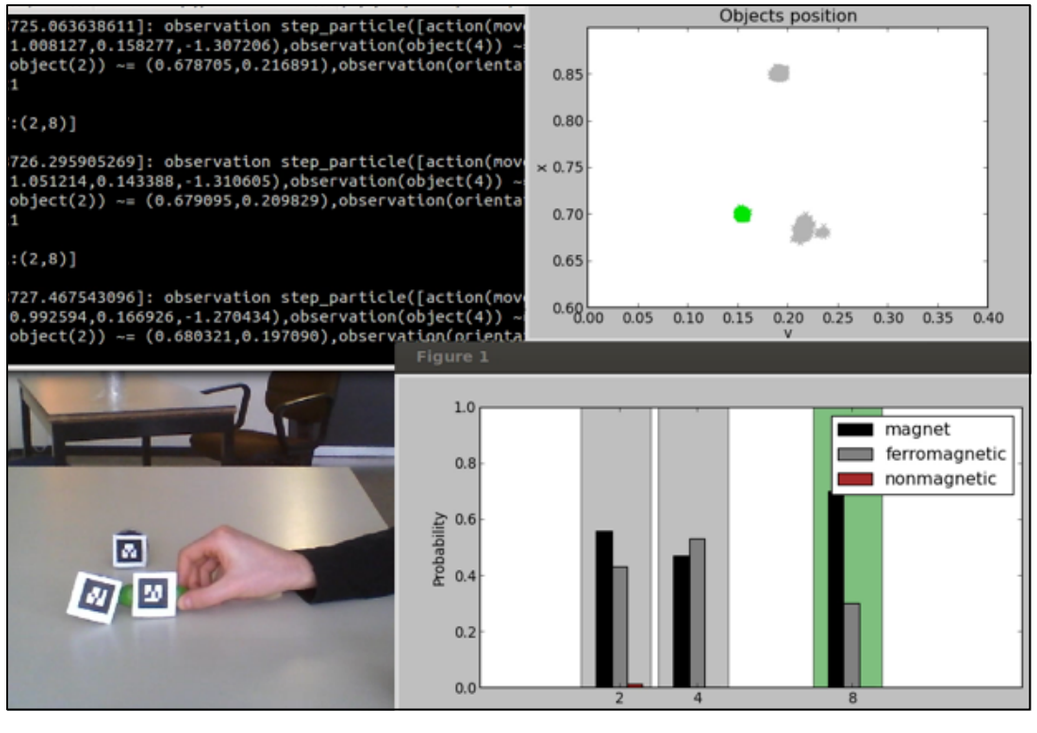
Social Network of Chats



Relational State Estimation over Time

Magnetism scenario

- object tracking
- category estimation from interactions

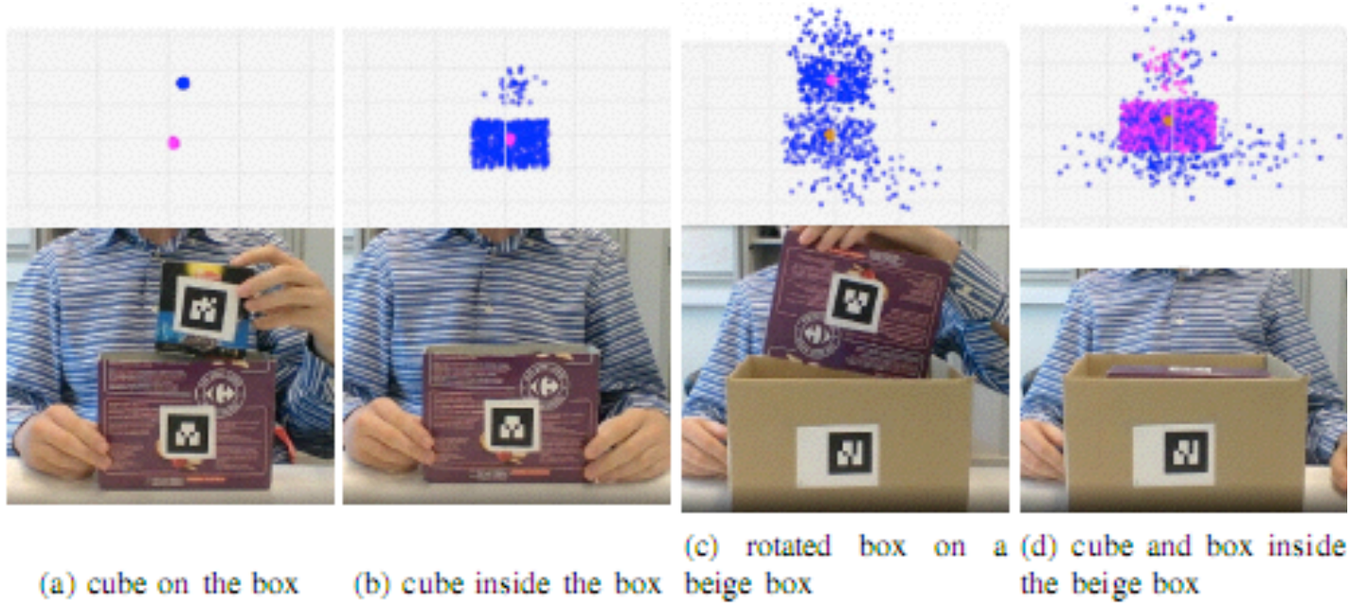


[Nitti et al, IROS 13]

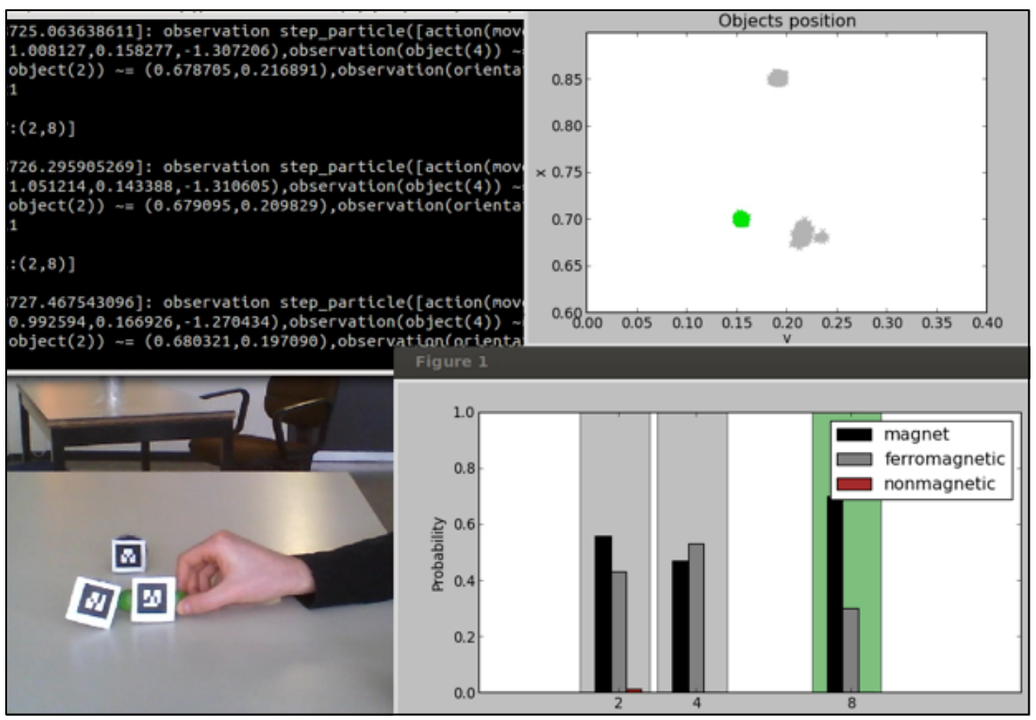
Relational State Estimation over Time

Magnetism scenario

- object tracking
- category estimation from interactions



(a) cube on the box (b) cube inside the box (c) rotated box on a beige box (d) cube and box inside the beige box



Box scenario

- object tracking even when invisible
- estimate spatial relations

[Nitti et al, IROS 13]

Distributional Clauses (DC)

- Discrete- and continuous-valued random variables

[Gutmann et al, TPLP 11; Nitti et al, IROS 13]

Distributional Clauses (DC)

- Discrete- and continuous-valued random variables

random variable with Gaussian distribution

```
length(Obj) ~ gaussian(6.0,0.45) :- type(Obj,glass).
```

[Gutmann et al, TPLP 11; Nitti et al, IROS 13]

Distributional Clauses (DC)

- Discrete- and continuous-valued random variables

```
length(Obj) ~ gaussian(6.0,0.45) :- type(Obj,glass).
stackable(OBot,OTop) :-
    ≃length(OBot) ≥ ≃length(OTop),
    ≃width(OBot) ≥ ≃width(OTop).
```

comparing values of random variables

[Gutmann et al, TPLP 11; Nitti et al, IROS 13]

Distributional Clauses (DC)

- Discrete- and continuous-valued random variables

```
length(Obj) ~ gaussian(6.0,0.45) :- type(Obj,glass).
stackable(OBot,OTop) :-
      ≃length(OBot) ≥ ≃length(OTop),
      ≃width(OBot) ≥ ≃width(OTop).
ontype(Obj,plate) ~ finite([0 : glass, 0.0024 : cup,
                            0 : pitcher, 0.8676 : plate,
                            0.0284 : bowl, 0 : serving,
                            0.1016 : none])
                        :- obj(Obj), on(Obj,O2), type(O2,plate).
```

random variable with discrete distribution

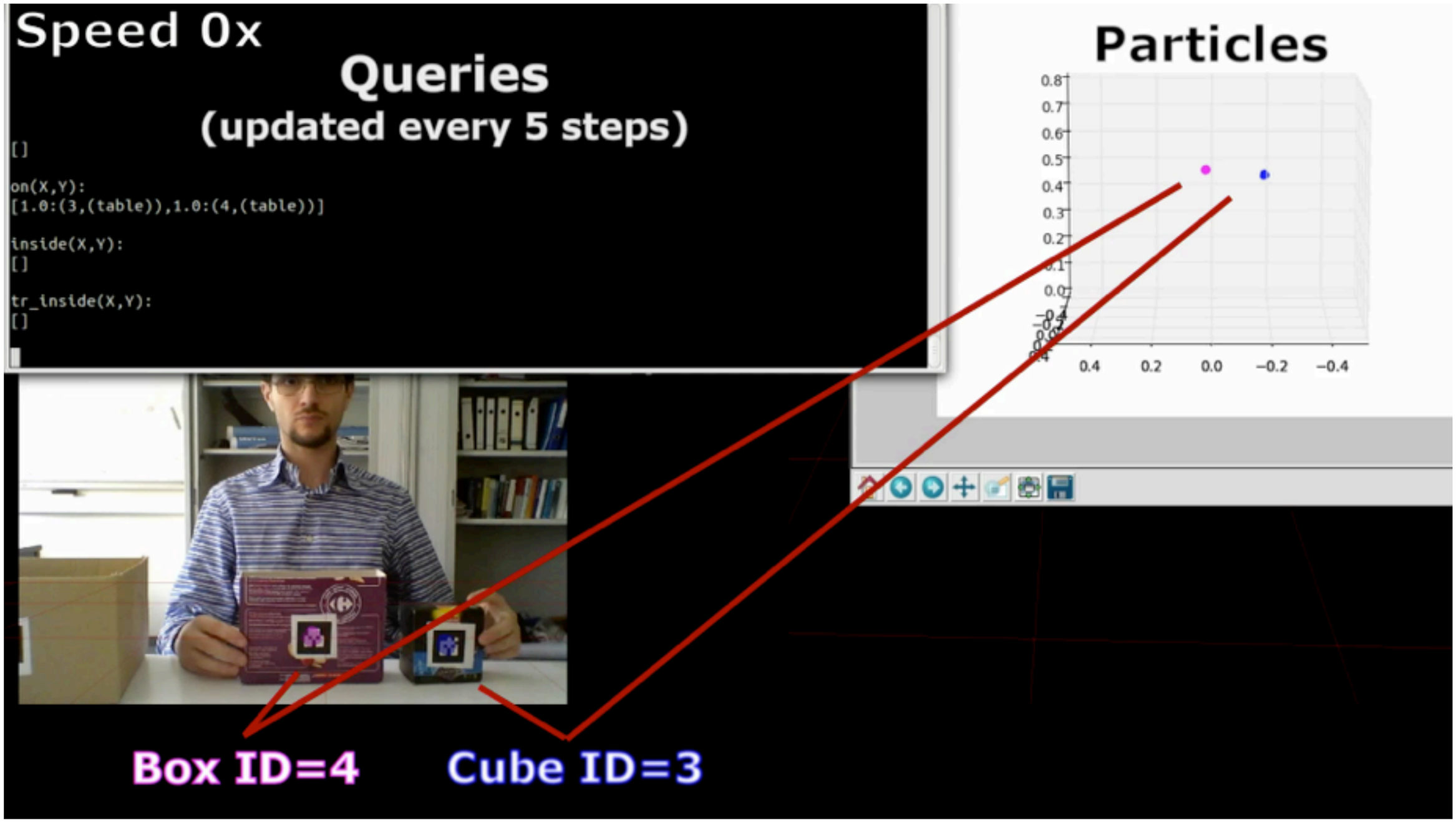
[Gutmann et al, TPLP 11; Nitti et al, IROS 13]

Distributional Clauses (DC)

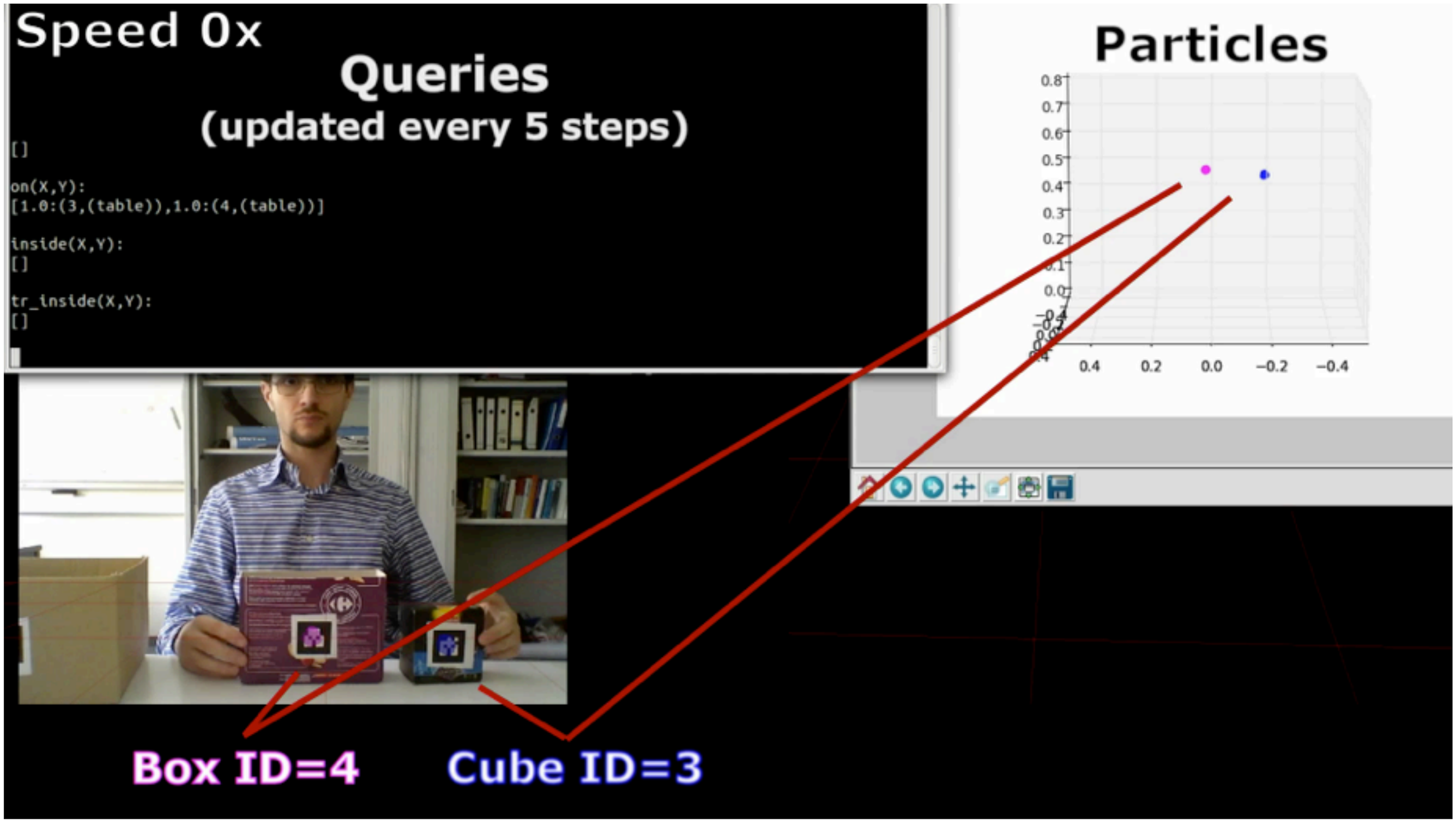
- Discrete- and continuous-valued random variables

```
length(Obj) ~ gaussian(6.0,0.45) :- type(Obj,glass).
stackable(OBot,OTop) :-
      ≃length(OBot) ≥ ≃length(OTop),
      ≃width(OBot) ≥ ≃width(OTop).
ontype(Obj,plate) ~ finite([0 : glass, 0.0024 : cup,
                            0 : pitcher, 0.8676 : plate,
                            0.0284 : bowl, 0 : serving,
                            0.1016 : none])
                    :- obj(Obj), on(Obj,O2), type(O2,plate).
```

[Gutmann et al, TPLP 11; Nitti et al, IROS 13]


Speed 0x
Queries
(updated every 5 steps)
[]
on(X,Y):
[1.0:(3,(table)),1.0:(4,(table))]
inside(X,Y):
[]
tr_inside(X,Y):
[]
Particles
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.4
0.2
0.0
−0.2
−0.4
Box ID=4
Cube ID=3

IROS 13

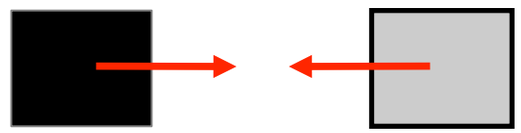

Speed 0x
Queries
(updated every 5 steps)
[]
on(X,Y):
[1.0:(3,(table)),1.0:(4,(table))]
inside(X,Y):
[]
tr_inside(X,Y):
[]
Particles
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.4
0.2
0.0
−0.2
−0.4
Box ID=4
Cube ID=3

IROS 13

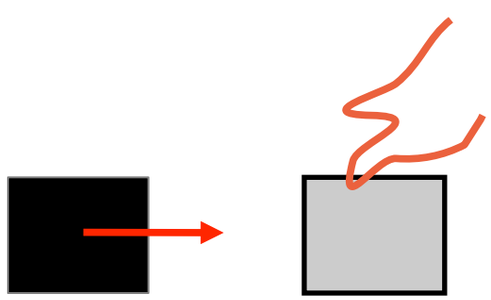
Magnetic scenario

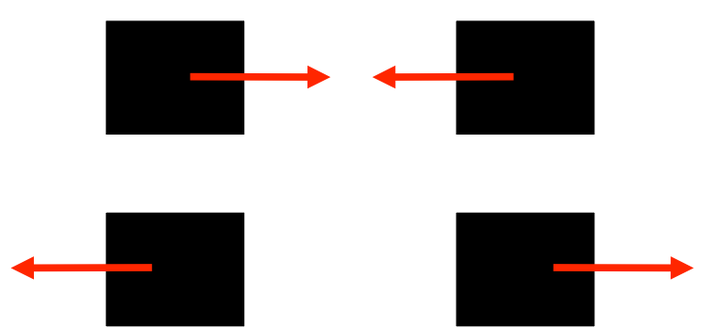
- 3 object types: magnetic, ferromagnetic, nonmagnetic

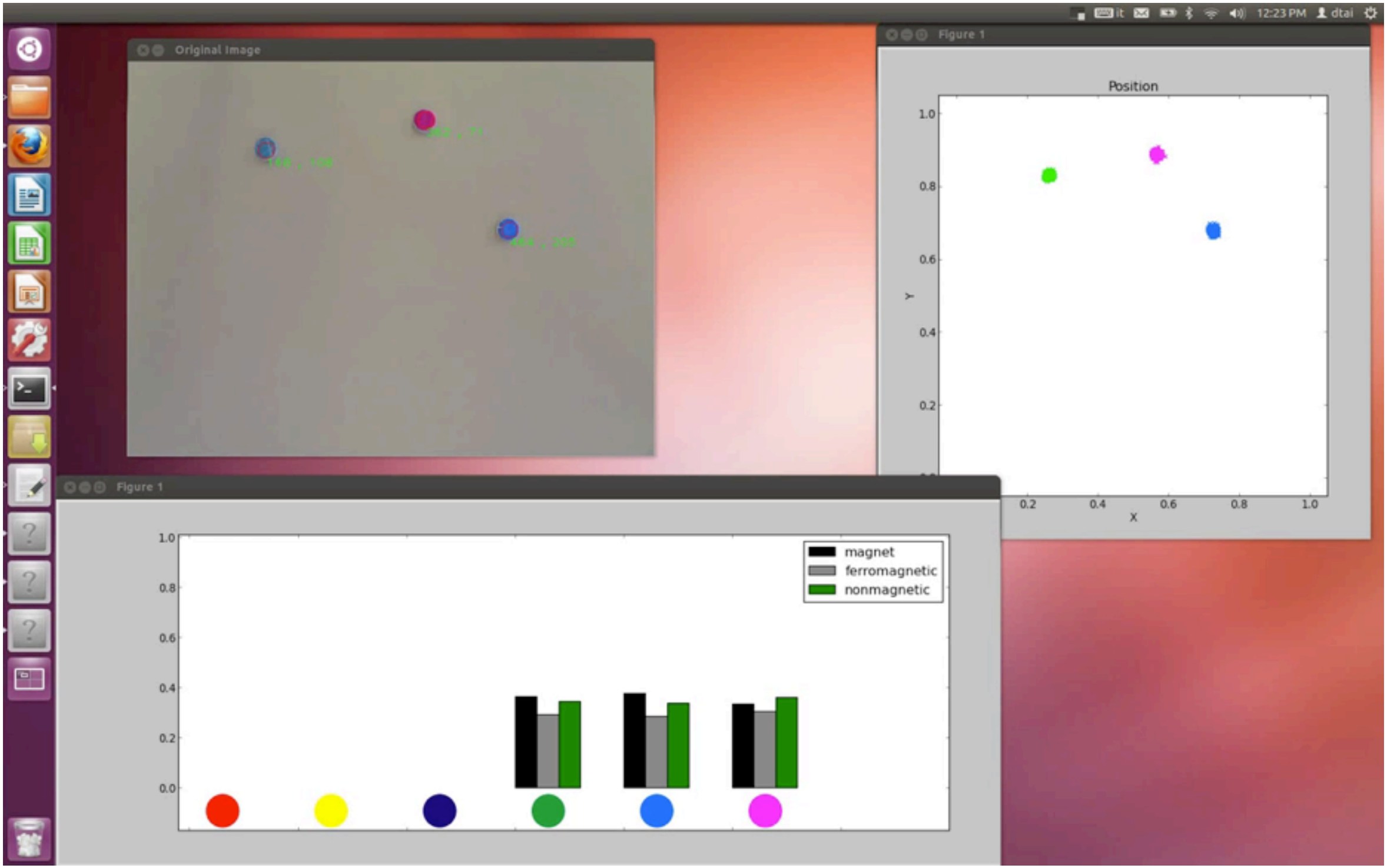
- Nonmagnetic objects do not interact
- A magnet and a ferromagnetic object attract each other

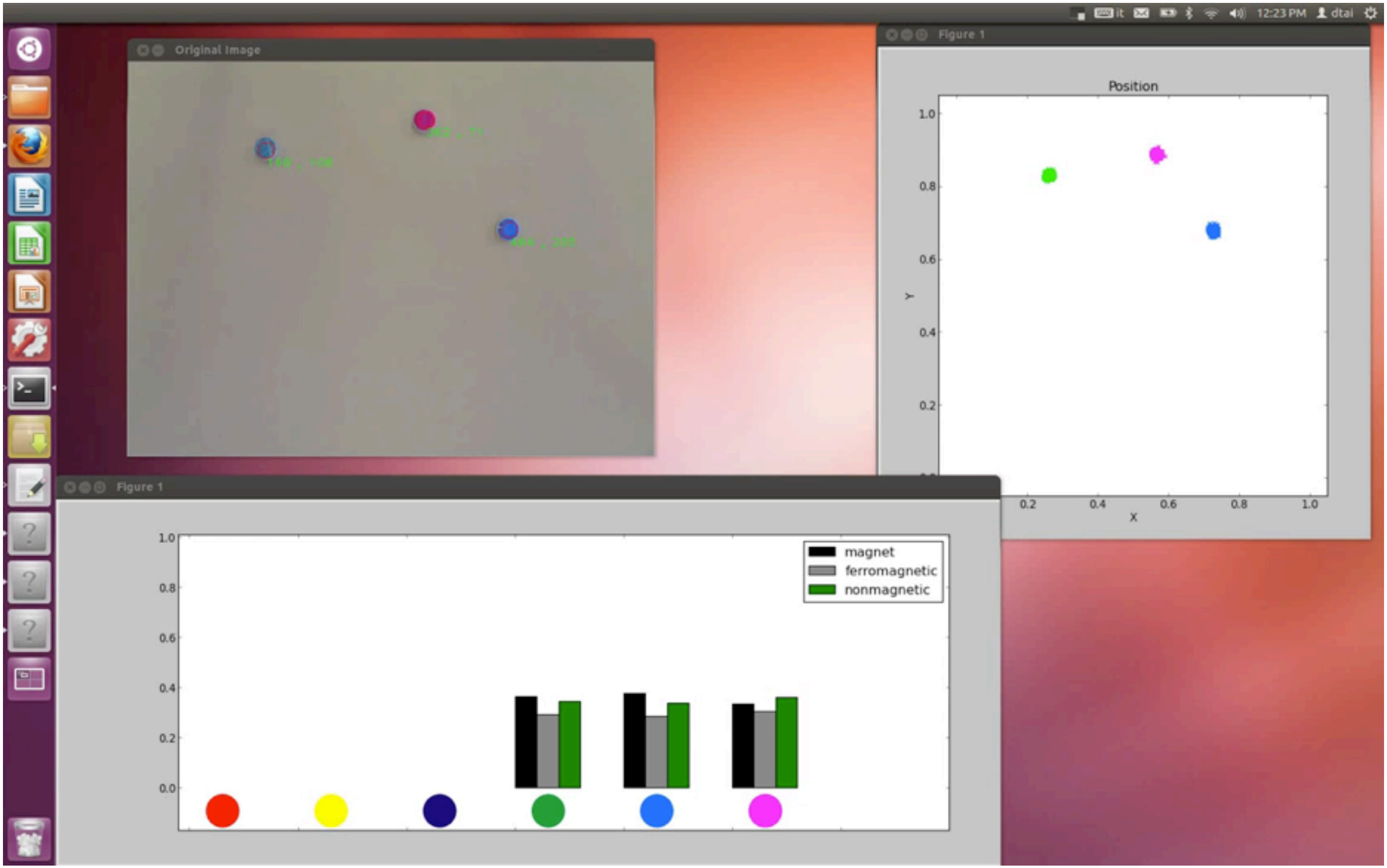


- Magnetic force that depends on the distance
- If an object is held magnetic force is compensated.






it
12:23 PM
dtai
Original Image
462 , 71
160 , 108
464 , 205
Figure 1
Position
1.0
0.8
0.6
0.4
0.2
Y
0.2
0.4
0.6
0.8
1.0
X
Figure 1
1.0
0.8
0.6
0.4
0.2
0.0
magnet
ferromagnetic
nonmagnetic


it
12:23 PM
dtai
Original Image
462 , 71
160 , 108
464 , 205
Figure 1
Position
1.0
0.8
0.6
0.4
0.2
Y
0.2
0.4
0.6
0.8
1.0
X
Figure 1
1.0
0.8
0.6
0.4
0.2
0.0
magnet
ferromagnetic
nonmagnetic

Magnetic scenario

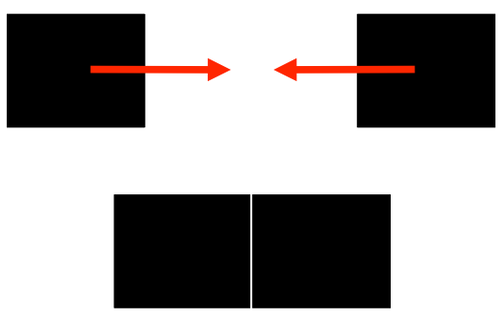
- 3 object types: magnetic, ferromagnetic, nonmagnetic

$$\text{type}(X)_t \sim \text{finite}([1/3\text{:magnet}, 1/3\text{:ferromagnetic}, 1/3\text{:nonmagnetic}]) \leftarrow \text{object}(X).$$

- 2 magnets attract or repulse

$$\text{interaction}(A,B)_t \sim \text{finite}([0.5\text{:attraction}, 0.5\text{:repulsion}]) \leftarrow \text{object}(A), \text{object}(B), A<B, \text{type}(A)_t = \text{magnet}, \text{type}(B)_t = \text{magnet}.$$

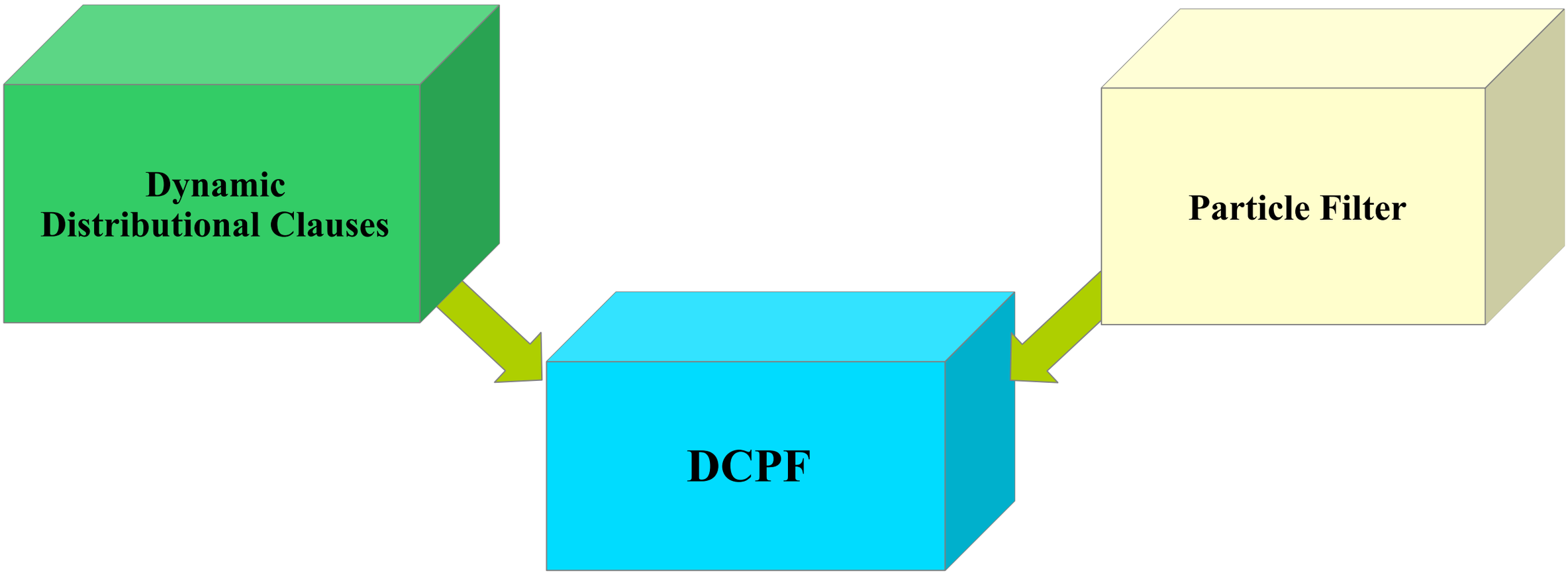
- Next position after attraction



$$\text{pos}(A)_{t+1} \sim \text{gaussian}(\text{middlepoint}(A,B)_t, \text{Cov}) \leftarrow \text{near}(A,B)_t, \text{not}(\text{held}(A)), \text{not}(\text{held}(B)), \text{interaction}(A,B)_t = \text{attr}, c/\text{dist}(A,B)_t^2 > \text{friction}(A)_t.$$

$$\text{pos}(A)_{t+1} \sim \text{gaussian}(\text{pos}(A)_t, \text{Cov}) \leftarrow \text{not}(\text{attraction}(A,B)).$$

DC Particle Filter (DCPF)



Goal

Flexible (relational) state representation

Fast inference (state estimation) in general models

"A particle filter for hybrid relational domains" IROS 2013
D. Nitti, T. De Laet, L. De Raedt

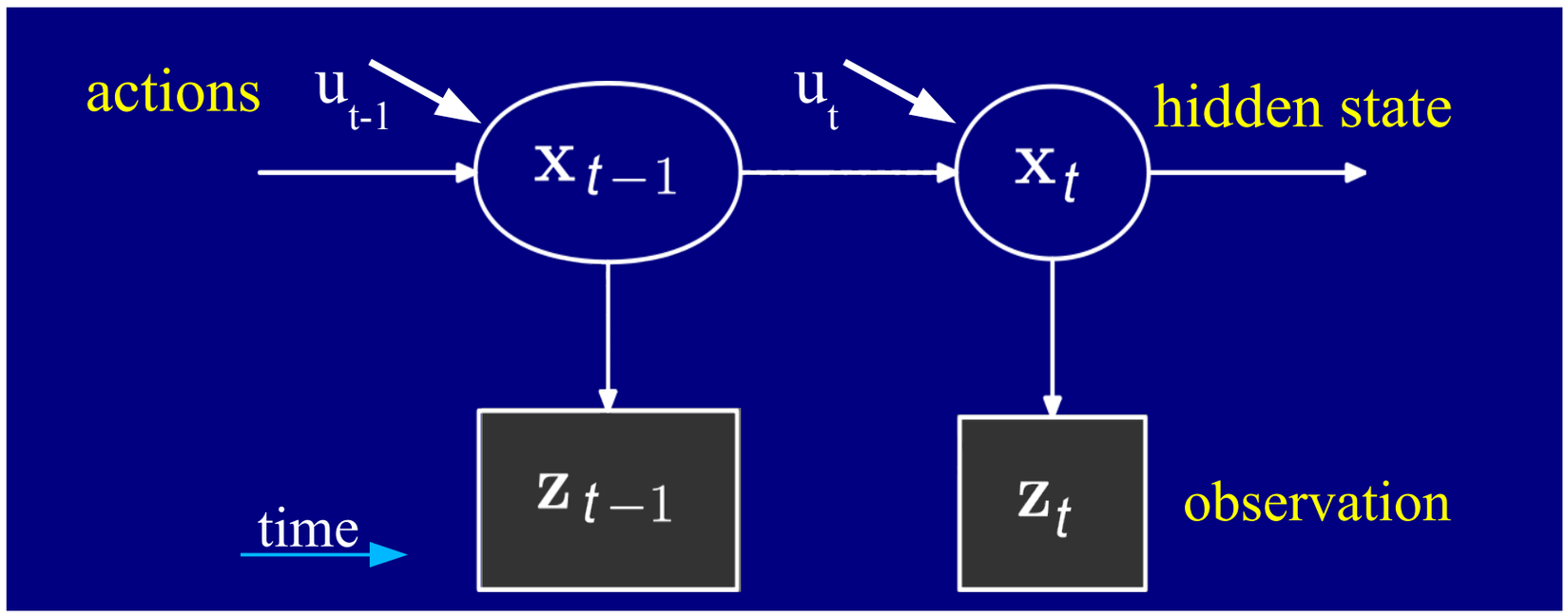
Dynamic Distributional Clauses

Prior distribution $p(x_0)$

State transition model $p(x_t|x_{t-1},u_t)$

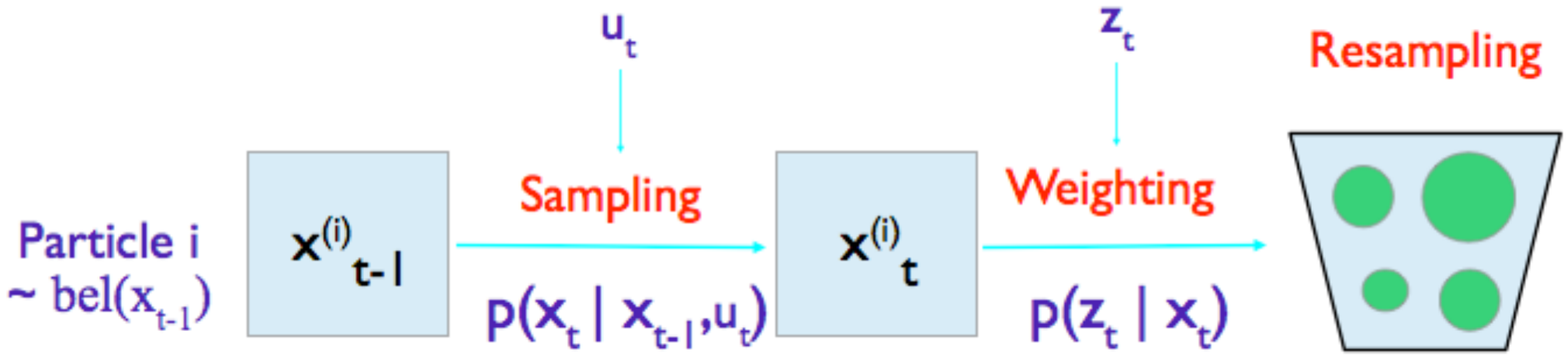
Measurement model $p(z_t|x_t)$

Other rules: $p(x'_t|x''_t)$



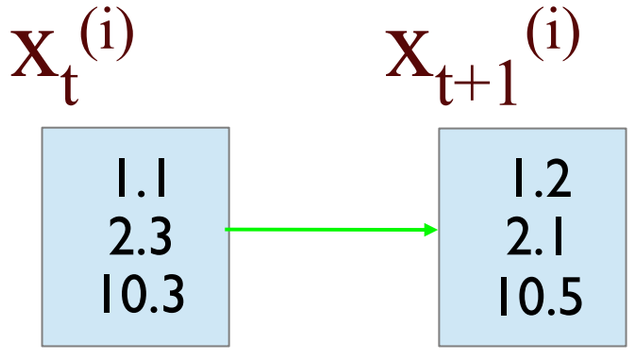
Particle Filter
(Sequential Monte Carlo)

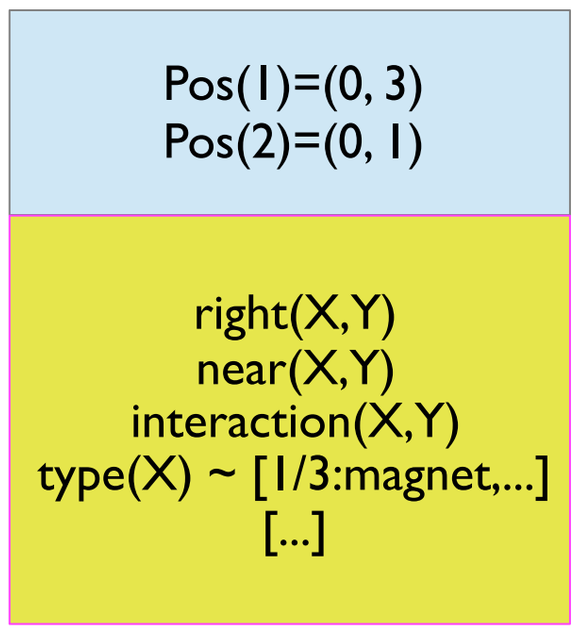
- Based on sampling → approximate inference
- Particles (samples) to represent bel(x_t)



Classical Particle Filter vs DCPF

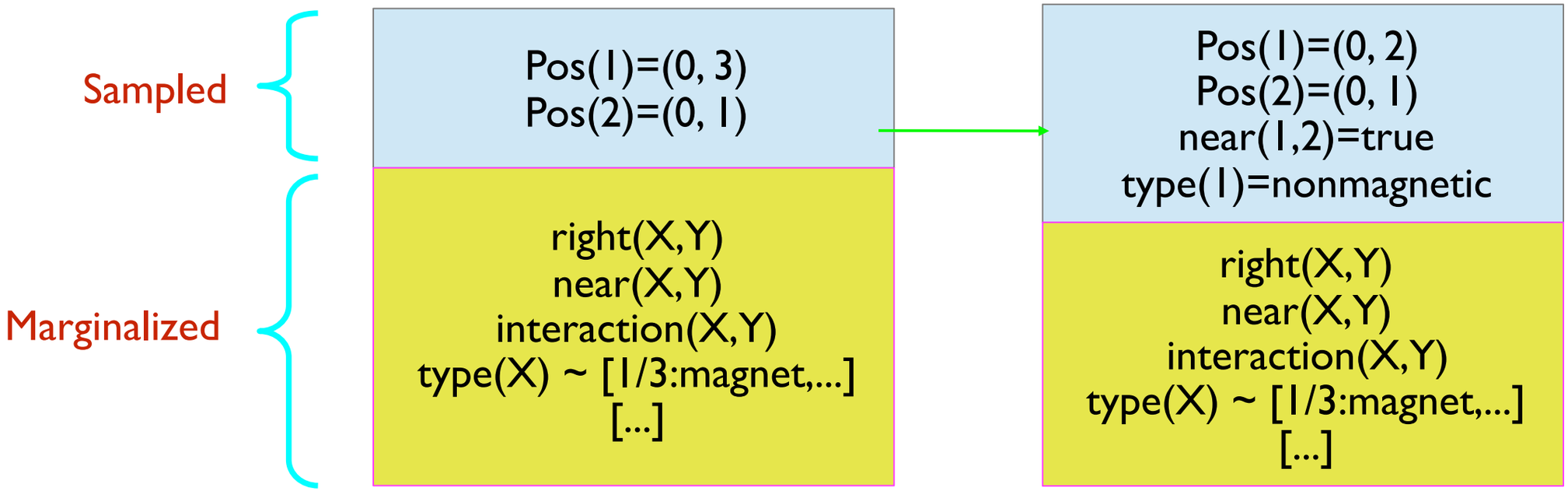
- Classical PF
  - Fixed set of random variables
  - Update the entire state
- DCPF
  - **Adaptive state** (particle): the number of facts / random variables can change over time
  - Particles are partial interpretations
  - Expressive language



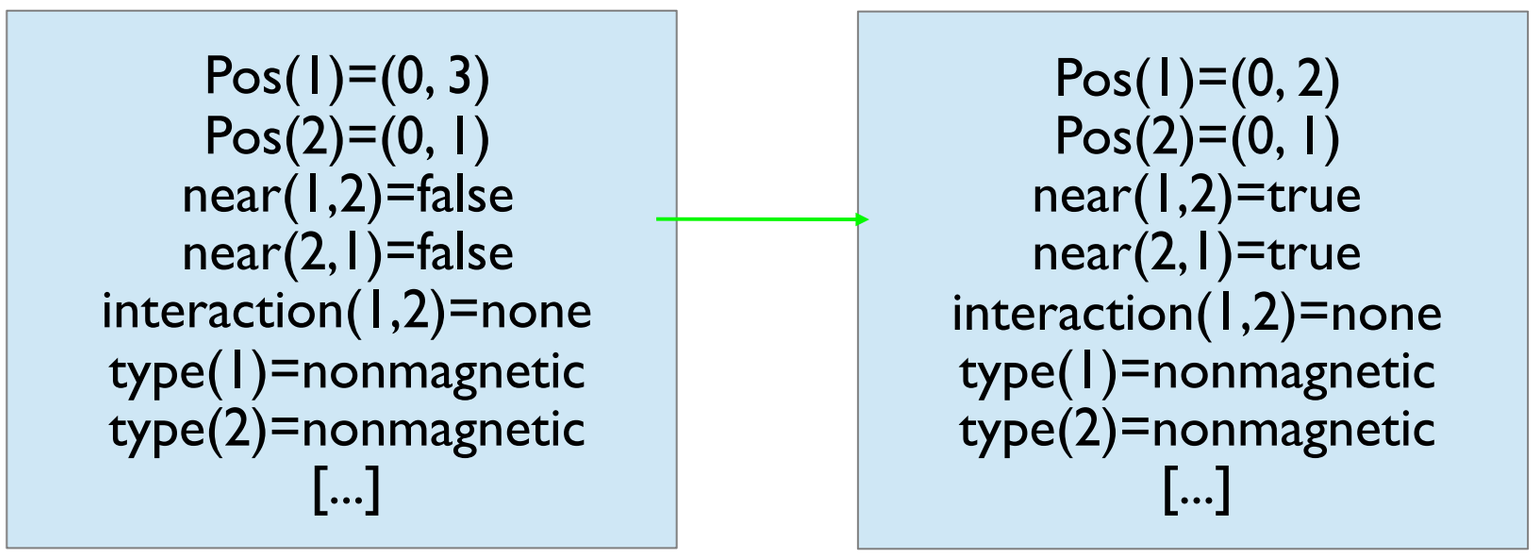


Optimized inference: partial state

Distributional Clauses Particle Filter (DCPF)

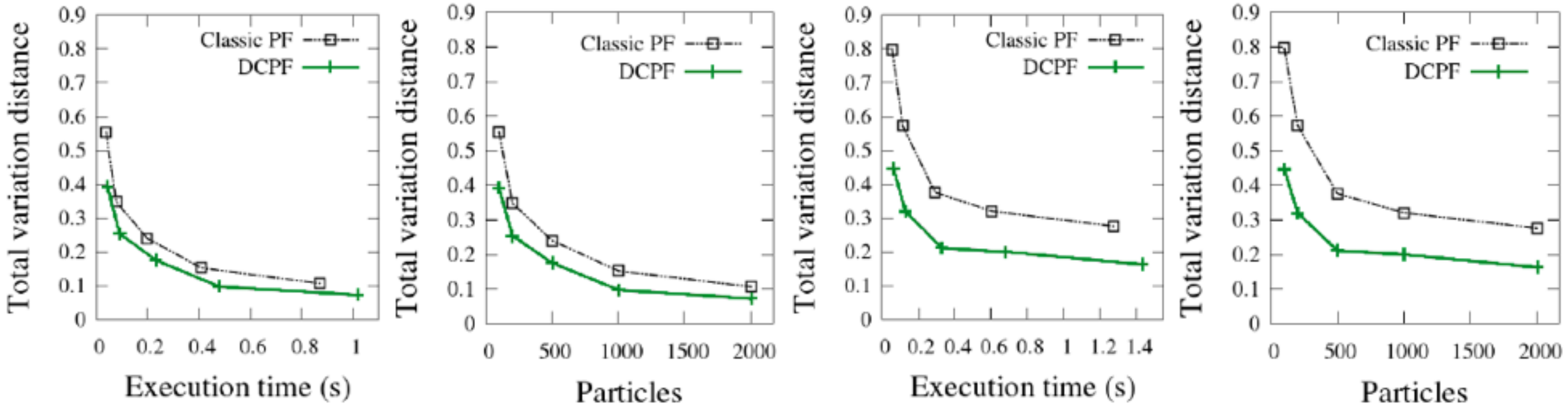


Classical particle filter



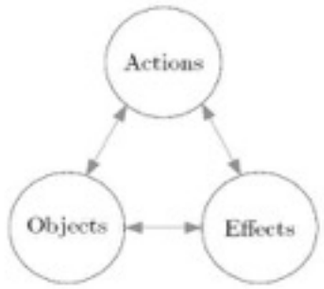
Experiments

- Particles are partial state, remaining variables are marginalized
- Better performance in bigger models



Learning relational affordances

Learn probabilistic model



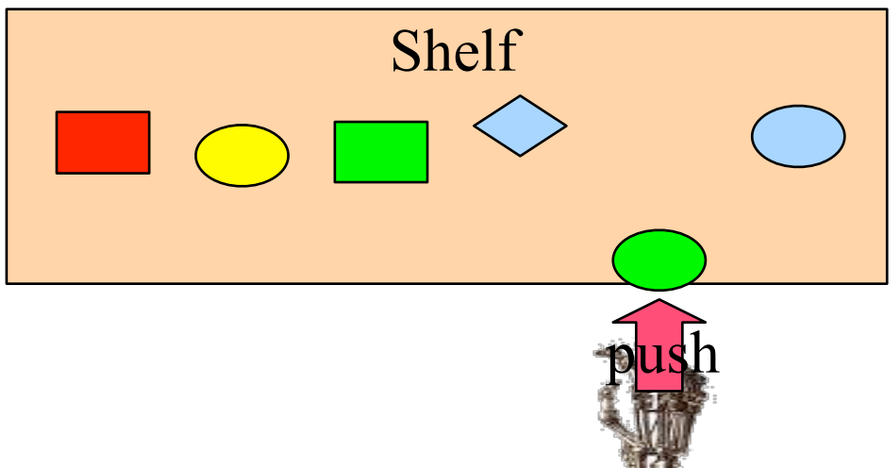
Inputs	Outputs	Function
(O, A)	E	Effect prediction
(O, E)	A	Action recognition/planning
(A, E)	O	Object recognition/selection

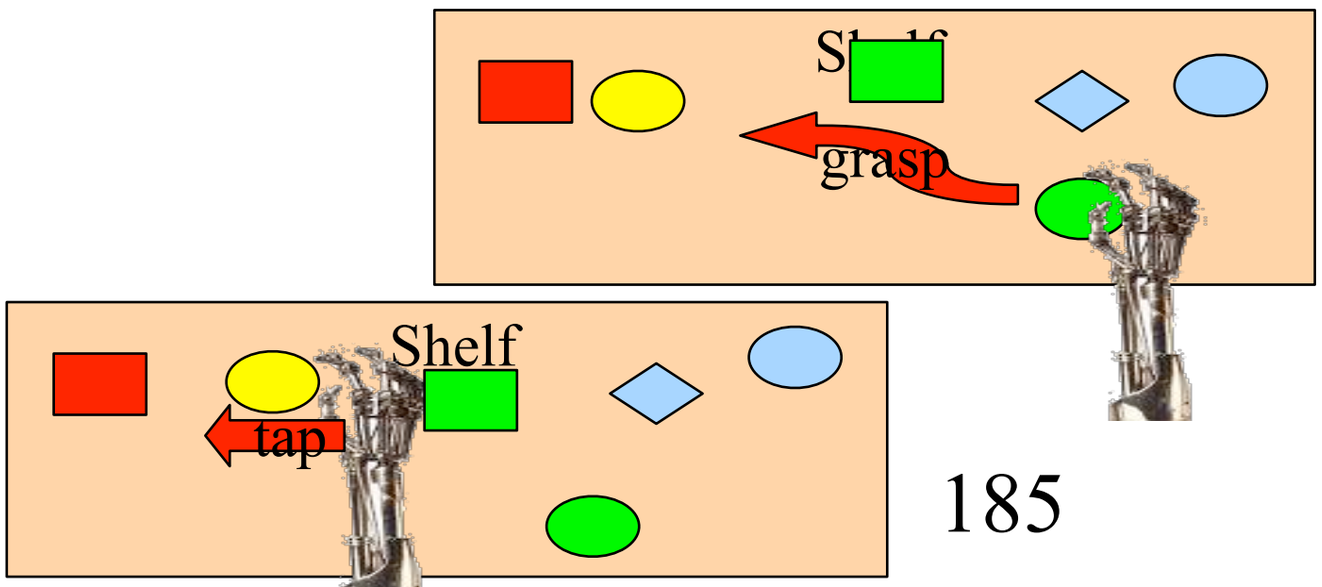
Learning relational affordances between two objects (learnt by experience)

From two object interactions

Generalize to N

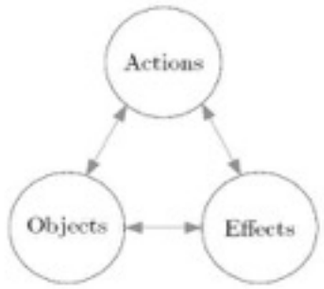
Moldovan et al. ICRA 12, 13, 14





Learning relational affordances

Learn probabilistic model



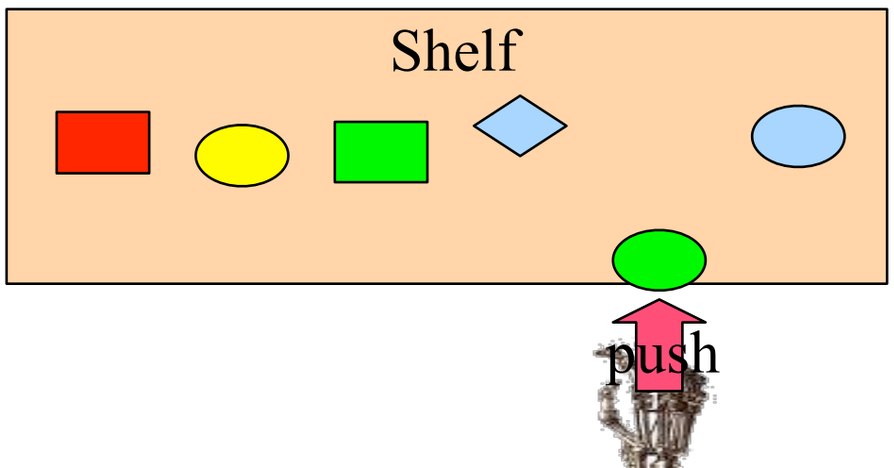
Inputs	Outputs	Function
(O, A)	E	Effect prediction
(O, E)	A	Action recognition/planning
(A, E)	O	Object recognition/selection

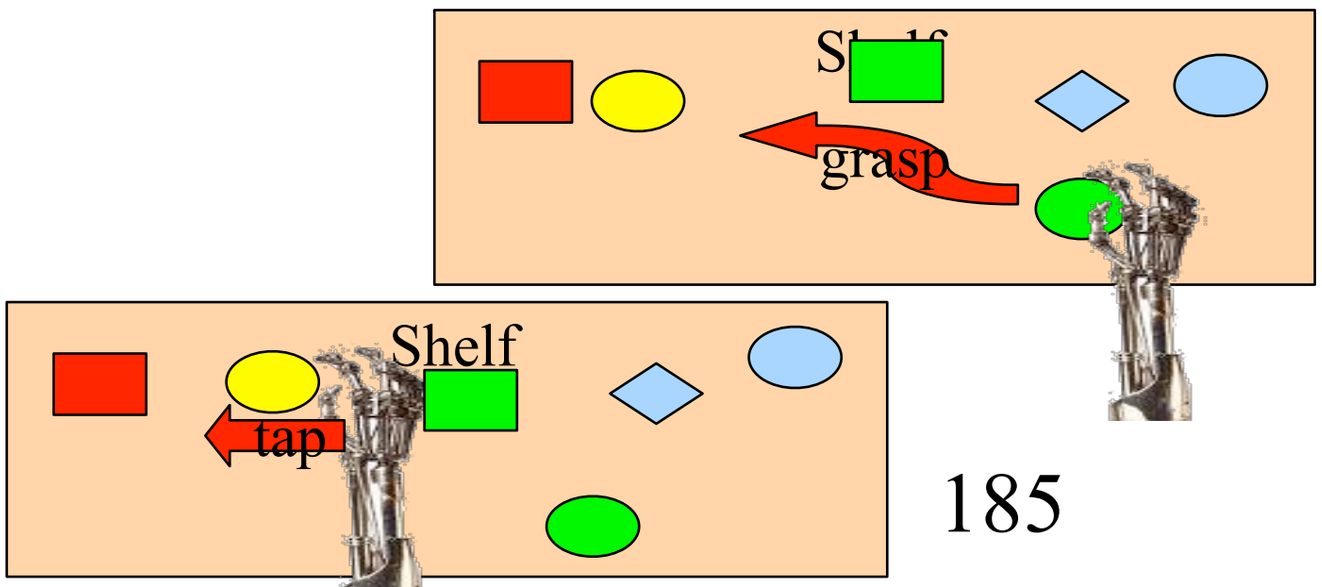
Learning relational affordances between two objects (learnt by experience)

From two object interactions

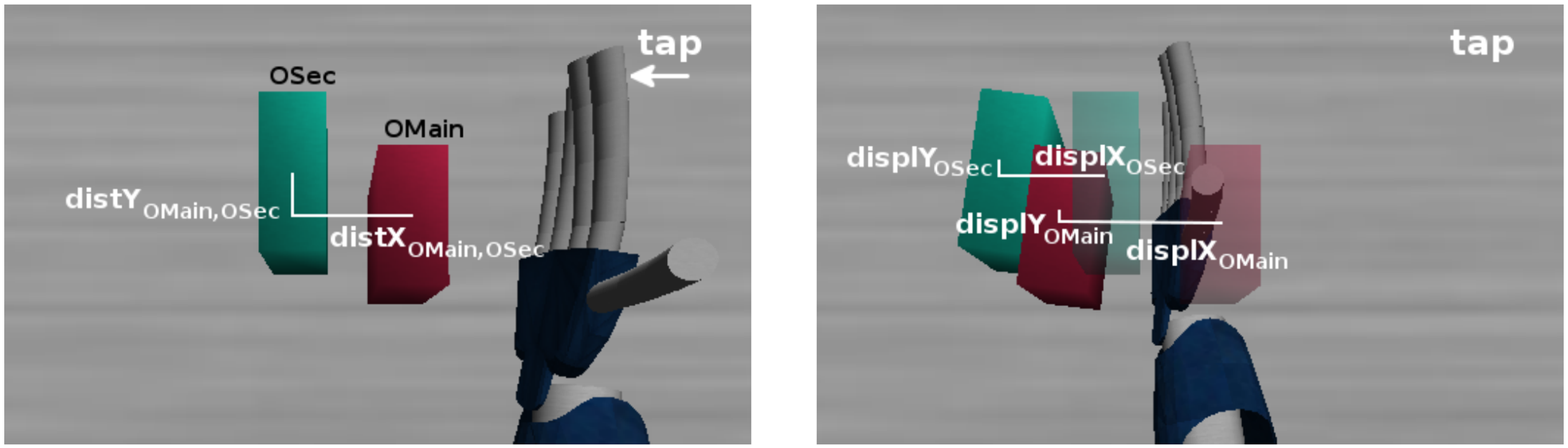
Generalize to N

Moldovan et al. ICRA 12, 13, 14





What is an affordance ?



Clip 8: Relational O before (l), and E after the action execution (r).

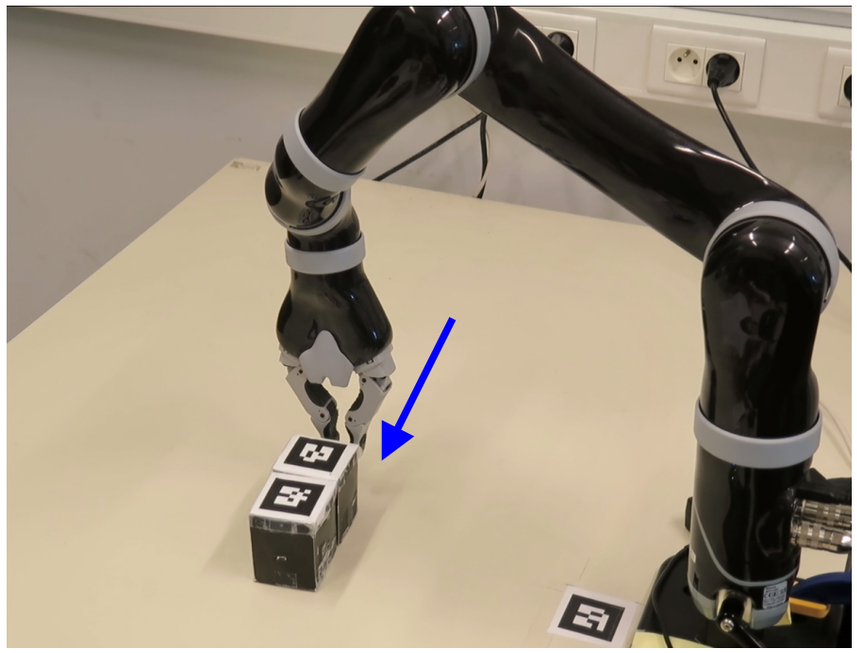
Table 1: Example collected O, A, E data for action in Figure 8

Object Properties	**Action**	**Effects**
$shape_{O_{Main}} : sprism$ $shape_{O_{Sec}} : sprism$ $distX_{O_{Main},O_{Sec}} : 6.94cm$ $distY_{O_{Main},O_{Sec}} : 1.90cm$	$tap(10)$	$displX_{O_{Main}} : 10.33cm$ $displY_{O_{Main}} : -0.68cm$ $displX_{O_{Sec}} : 7.43cm$ $displY_{O_{Sec}} : -1.31cm$

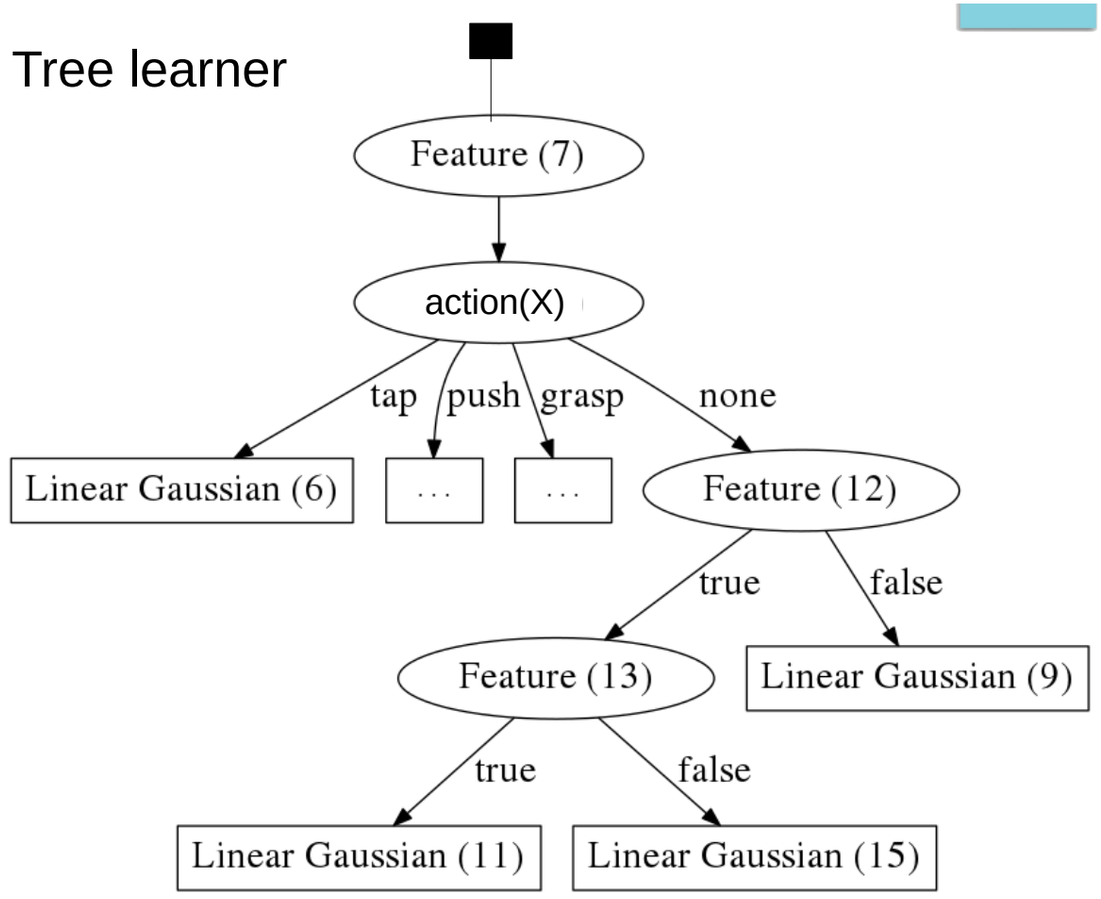
- Formalism — related to STRIPS but models delta
- but also joint probability model over A, E, O

Relational Affordance Learning

- **Learning the Structure of Dynamic Hybrid Relational Models**
  Nitti, Ravkic, et al. ECAI 2016
  - Captures relations/affordances
  - Suited to learn affordances in
    robotics set-up, continuous and discrete variables
  - Planning in hybrid robotics domain



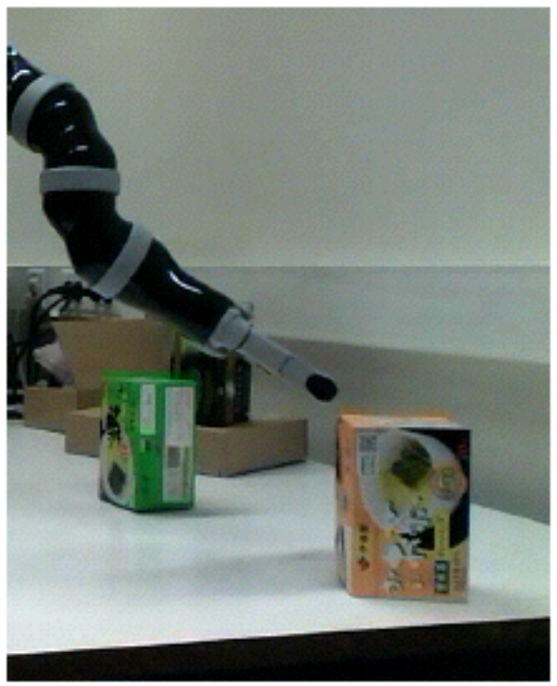
DDC Tree learner

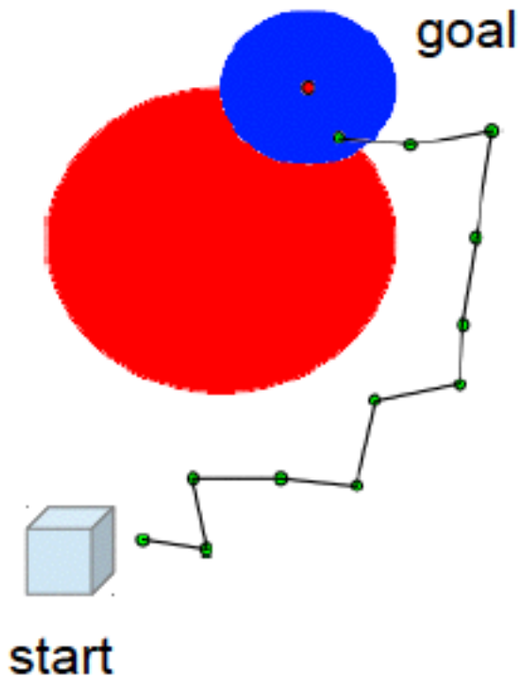


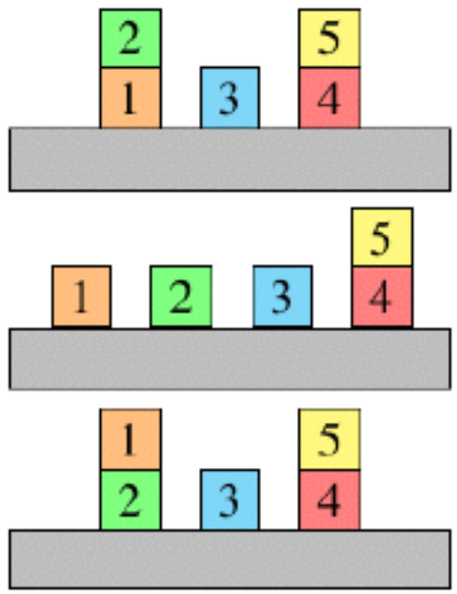
Planning

- Main task: probabilistic planning

  Find the best action to achieve the goal
- Discrete + continuous + relational representation



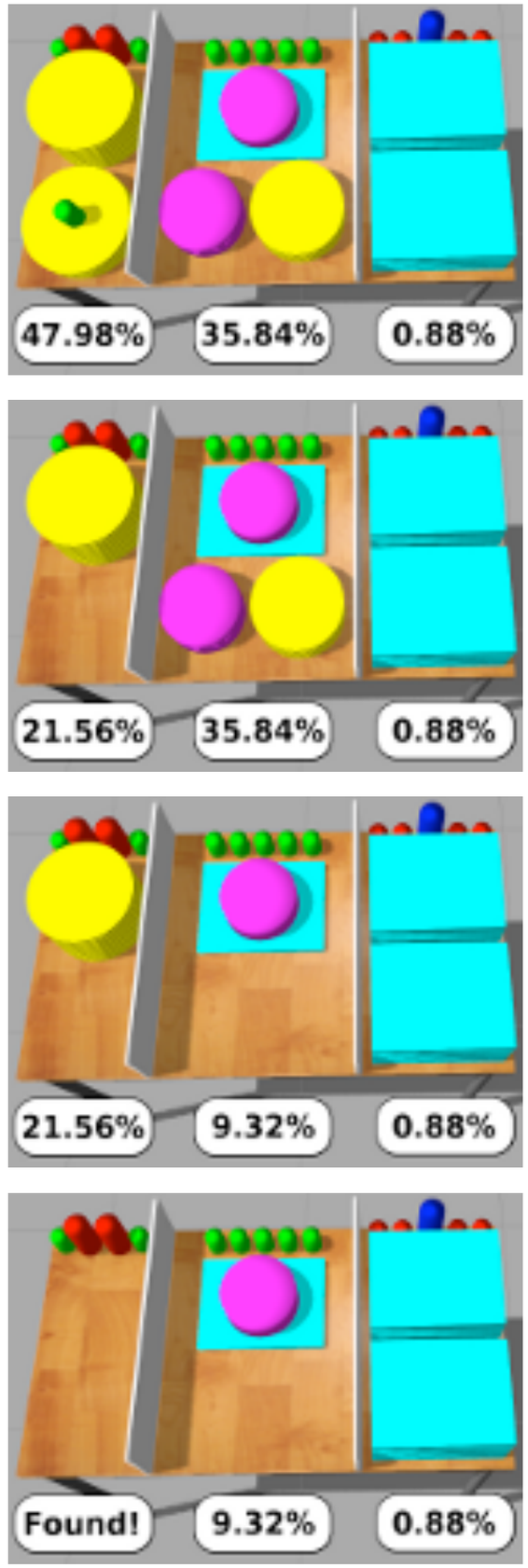




[Nitti et al ECML 15]

Occluded Object Search

- How to achieve a specific configuration of objects on the shelf?
- Where's the orange mug?
- Where's something to serve soup in?
- Models of objects and their spatial arrangement



[Moldovan *et al. 14*]

ProbLog for activity recognition from video

CAVIAR-INRIA human activity dataset

28 videos
≈ 26.500 frames

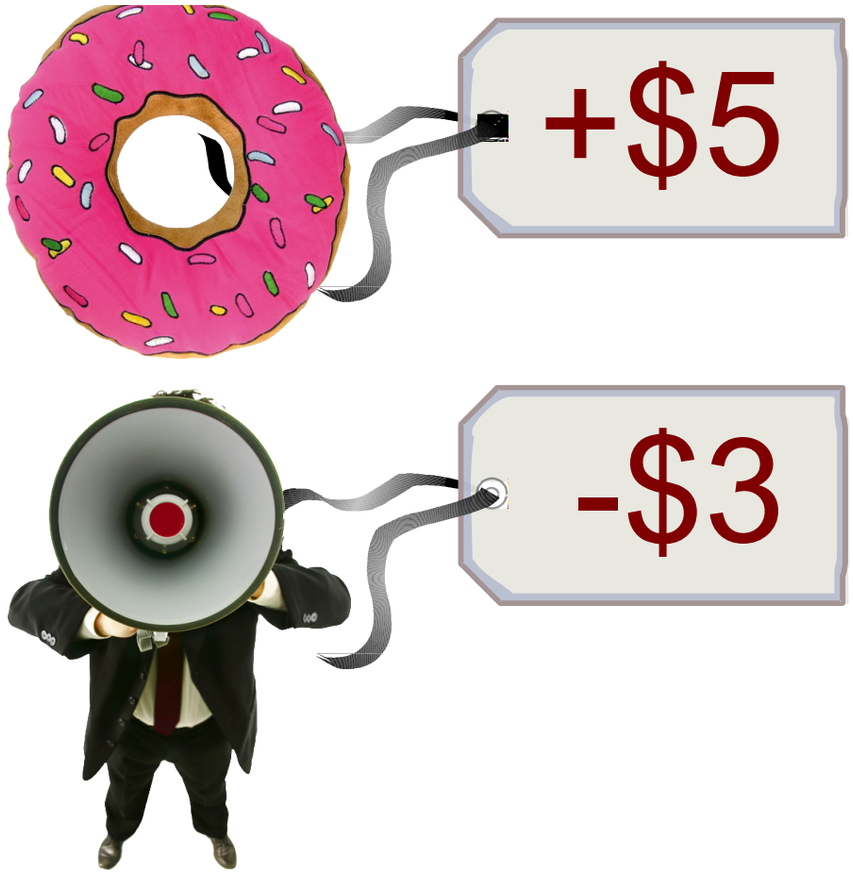
- Separation between low-level events (LLE) and high-level events (HLE)
  - LLE: *walking, running, active, inactive, abrupt*
  - HLE: *meeting, moving, fighting, leaving_object*
- Probabilistic Logic approach: *Event Calculus in ProbLog* (Prob-EC) to infer the high-level events from an **algebra** of low-level events.
- Example:

$$initiatedAt(fighting(P_1, P_2) = true, T) \leftarrow$$
$$happensAt(abrupt(P_1), T),$$
$$holdsAt(close(P_1, P_2, 44) = true, T),$$
$$not\ happensAt(inactive(P_2), T).$$

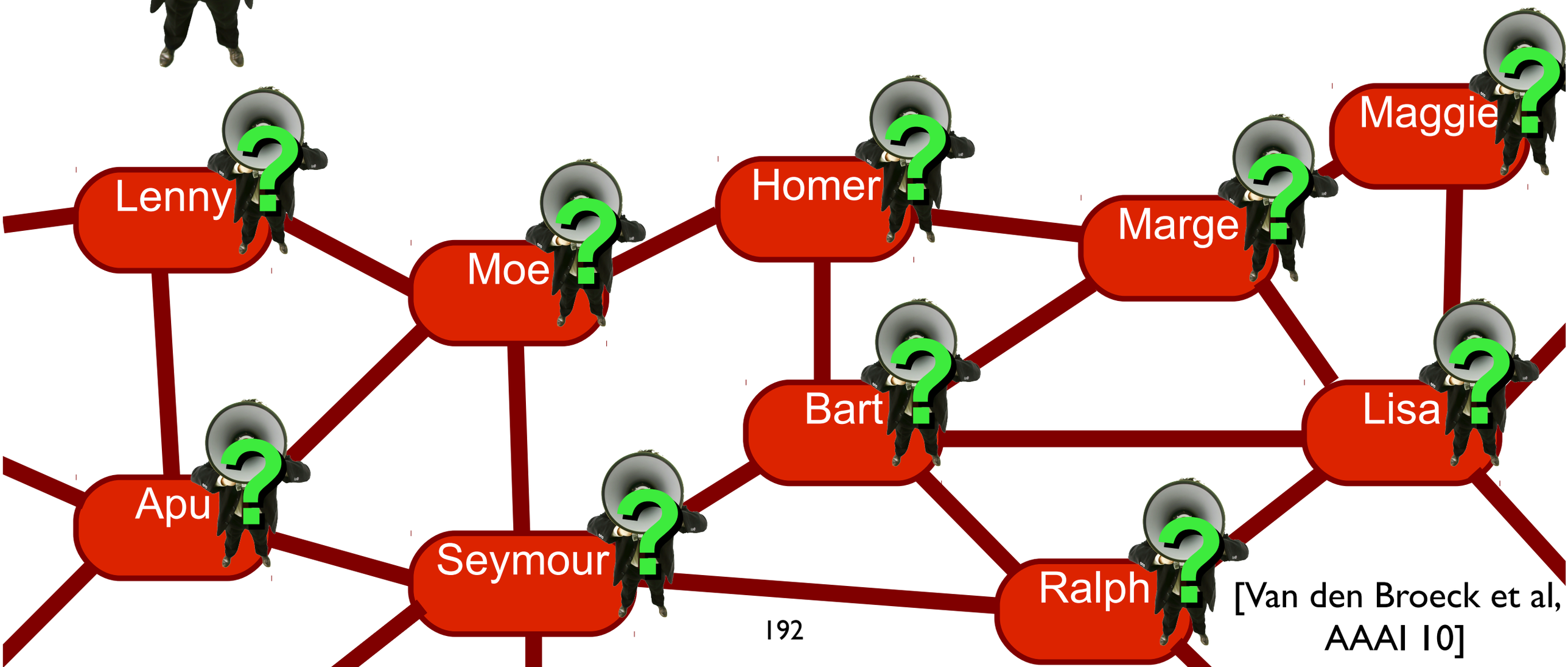
[Skarlatidis et al, TPLP 13]

Decisions

Viral Marketing

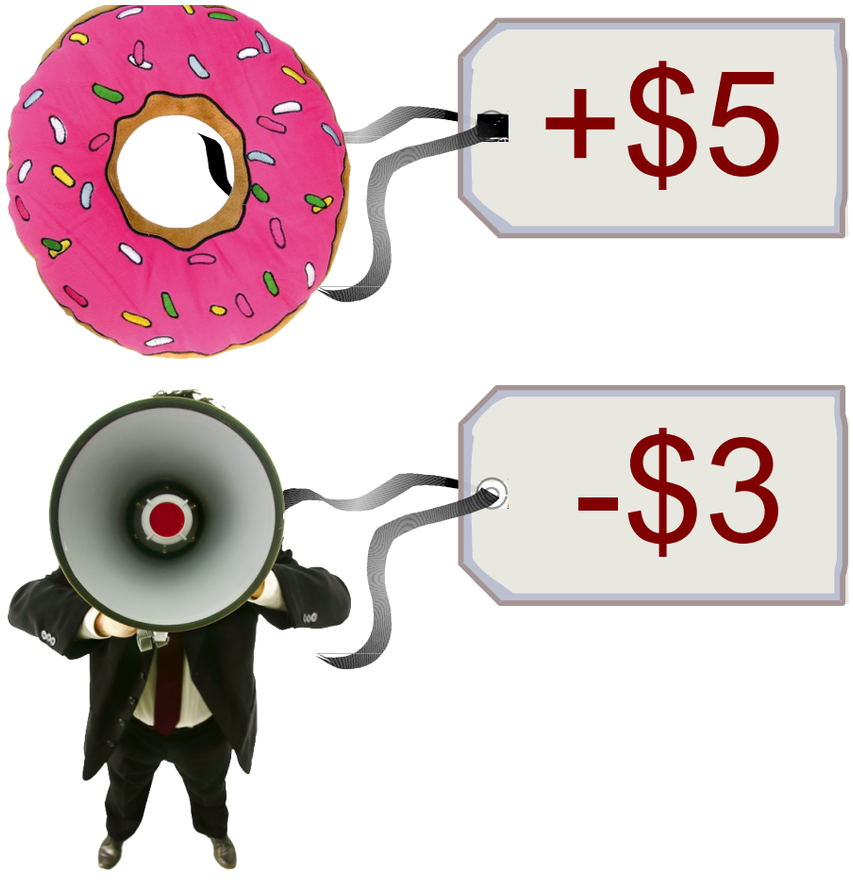


Which advertising strategy maximizes expected profit?

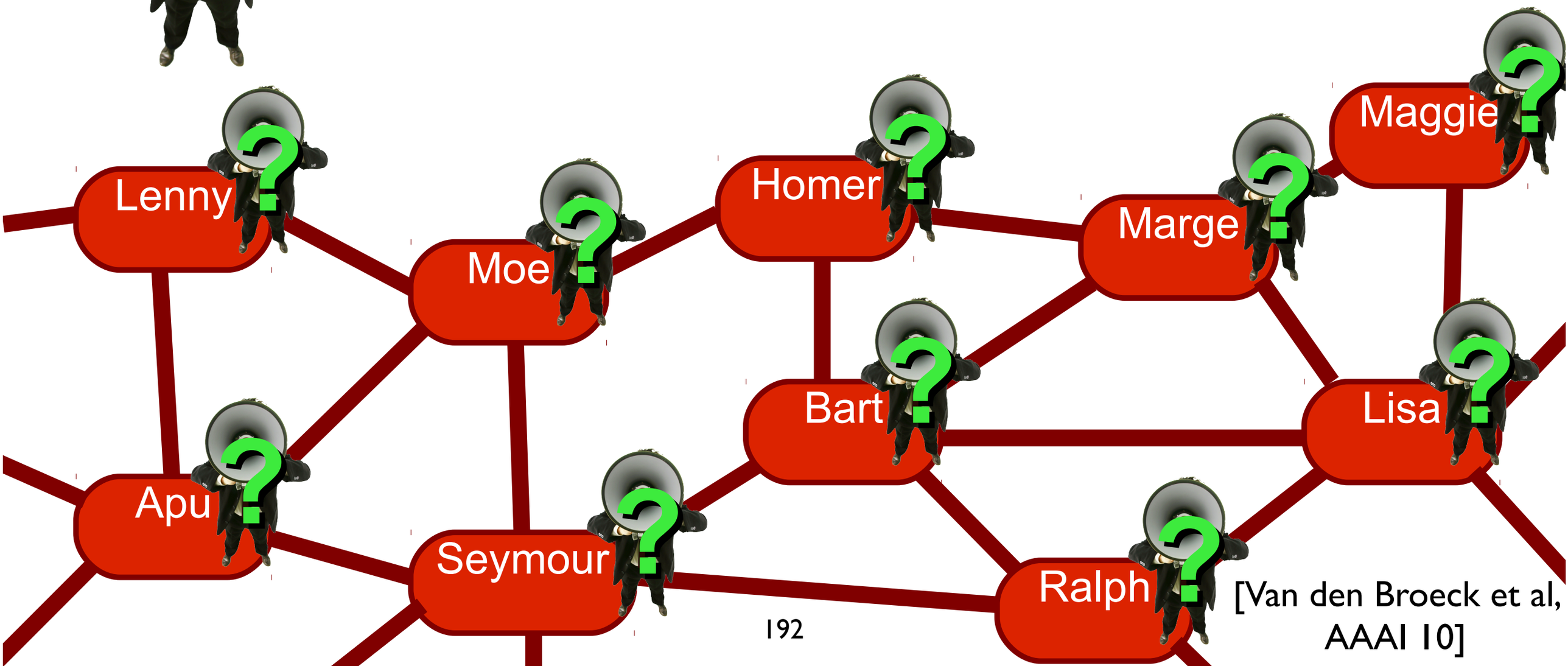


[Van den Broeck et al, AAAI 10]

Viral Marketing

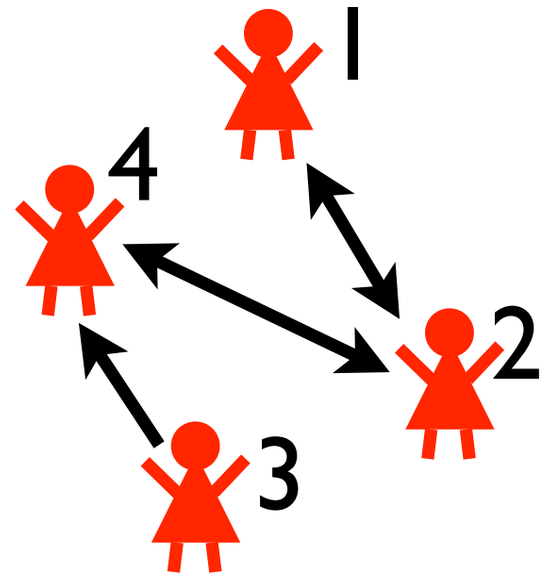


decide truth values of some atoms



[Van den Broeck et al, AAAI 10]

DTProbLog



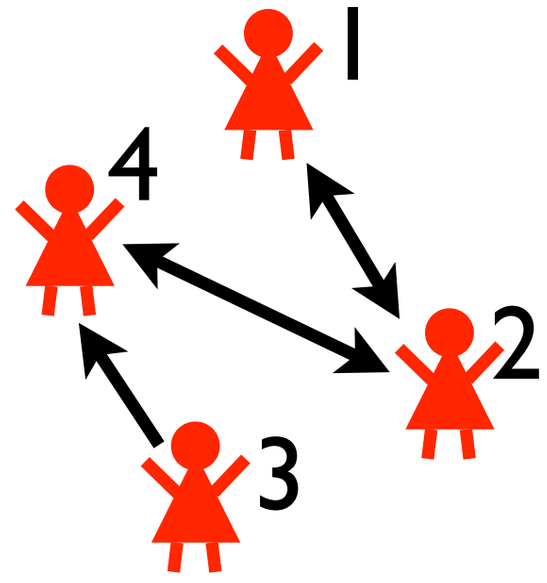

```
person(1).
person(2).
person(3).
person(4).
```

```
friend(1,2).
friend(2,1).
friend(2,4).
friend(3,4).
friend(4,2).
```

DTProbLog

```
? :: marketed(P) :- person(P).
```

decision fact: true or false?




```
person(1).
person(2).
person(3).
person(4).
```

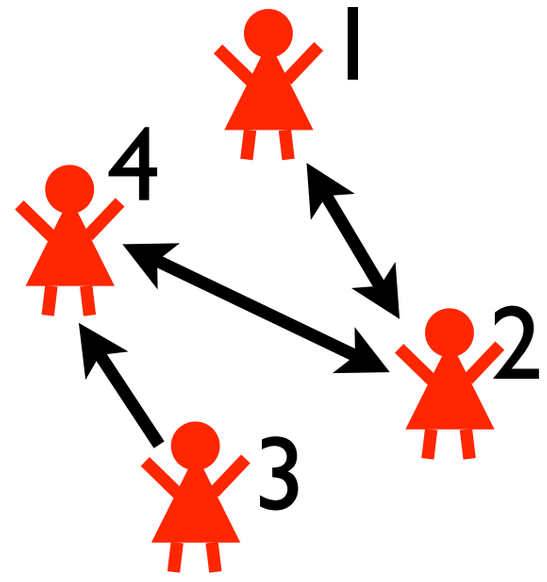
```
friend(1,2).
friend(2,1).
friend(2,4).
friend(3,4).
friend(4,2).
```

DTProbLog

```
? :: marketed(P) :- person(P).
```

```
0.3 :: buy_trust(X,Y) :- friend(X,Y).
0.2 :: buy_marketing(P) :- person(P).

buys(X) :- friend(X,Y), buys(Y), buy_trust(X,Y).
buys(X) :- marketed(X), buy_marketing(X).
```

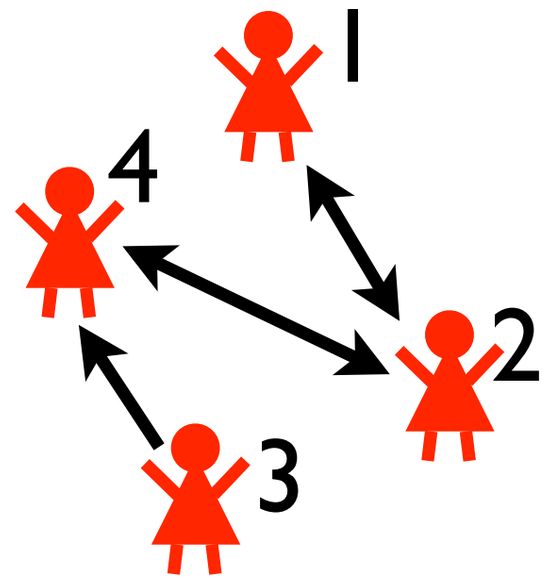
probabilistic facts
+ logical rules




```
person(1).
person(2).
person(3).
person(4).
```

```
friend(1,2).
friend(2,1).
friend(2,4).
friend(3,4).
friend(4,2).
```

DTProbLog

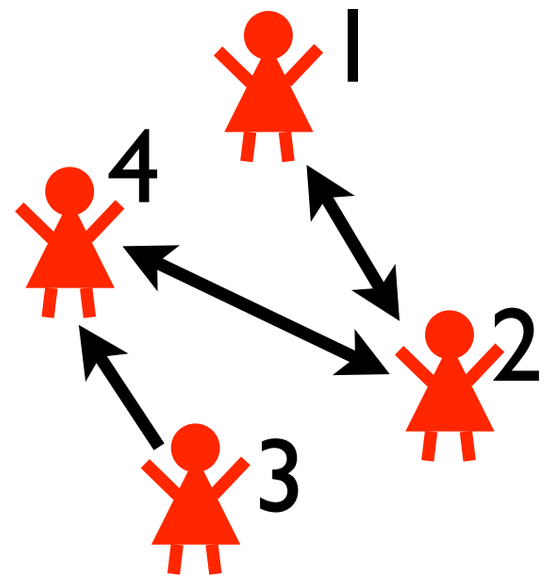



```
? :: marketed(P) :- person(P).

0.3 :: buy_trust(X,Y) :- friend(X,Y).
0.2 :: buy_marketing(P) :- person(P).

buys(X) :- friend(X,Y), buys(Y), buy_trust(X,Y).
buys(X) :- marketed(X), buy_marketing(X).

buys(P) => 5 :- person(P).
marketed(P) => -3 :- person(P).
```

```
person(1).
person(2).
person(3).
person(4).

friend(1,2).
friend(2,1).
friend(2,4).
friend(3,4).
friend(4,2).
```

utility facts: cost/reward if true

DTProbLog

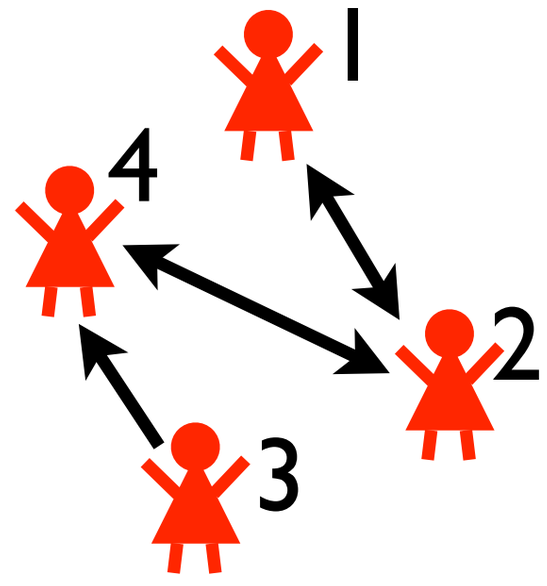



```
? :: marketed(P) :- person(P).

0.3 :: buy_trust(X,Y) :- friend(X,Y).
0.2 :: buy_marketing(P) :- person(P).

buys(X) :- friend(X,Y), buys(Y), buy_trust(X,Y).
buys(X) :- marketed(X), buy_marketing(X).

buys(P) => 5 :- person(P).
marketed(P) => -3 :- person(P).
```

```
person(1).
person(2).
person(3).
person(4).

friend(1,2).
friend(2,1).
friend(2,4).
friend(3,4).
friend(4,2).
```

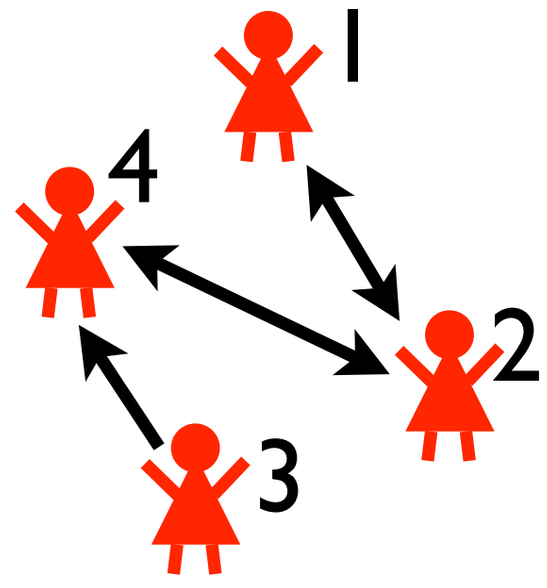
DTProbLog

```
? :: marketed(P) :- person(P).

0.3 :: buy_trust(X,Y) :- friend(X,Y).
0.2 :: buy_marketing(P) :- person(P).

buys(X) :- friend(X,Y), buys(Y), buy_trust(X,Y).
buys(X) :- marketed(X), buy_marketing(X).

buys(P) => 5 :- person(P).
marketed(P) => -3 :- person(P).
```




```
person(1).
person(2).
person(3).
person(4).

friend(1,2).
friend(2,1).
friend(2,4).
friend(3,4).
friend(4,2).
```

DTProbLog

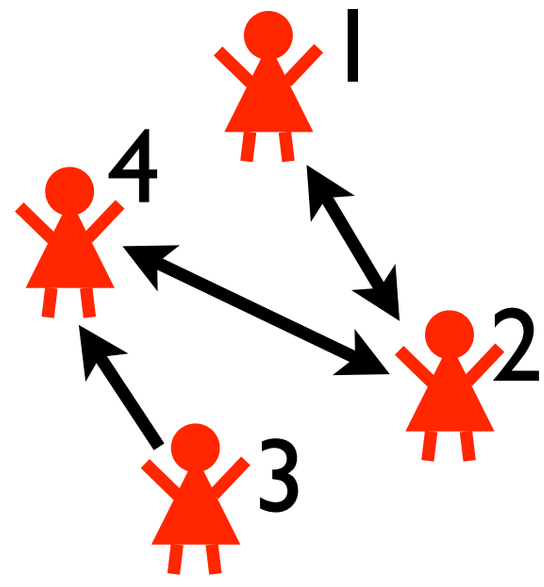



```
? :: marketed(P) :- person(P).

0.3 :: buy_trust(X,Y) :- friend(X,Y).
0.2 :: buy_marketing(P) :- person(P).

buys(X) :- friend(X,Y), buys(Y), buy_trust(X,Y).
buys(X) :- marketed(X), buy_marketing(X).

buys(P) => 5 :- person(P).
marketed(P) => -3 :- person(P).
```

```
person(1).
person(2).
person(3).
person(4).

friend(1,2).
friend(2,1).
friend(2,4).
friend(3,4).
friend(4,2).
```

```
marketed(1)        marketed(3)
```

DTProbLog

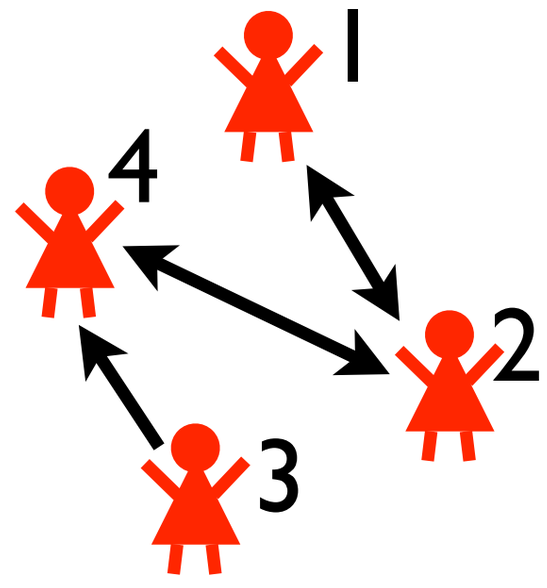



```
? :: marketed(P) :- person(P).

0.3 :: buy_trust(X,Y) :- friend(X,Y).
0.2 :: buy_marketing(P) :- person(P).

buys(X) :- friend(X,Y), buys(Y), buy_trust(X,Y).
buys(X) :- marketed(X), buy_marketing(X).

buys(P) => 5 :- person(P).
marketed(P) => -3 :- person(P).
```

```
person(1).
person(2).
person(3).
person(4).

friend(1,2).
friend(2,1).
friend(2,4).
friend(3,4).
friend(4,2).
```

```
marketed(1)        marketed(3)

    bt(2,1)   bt(2,4)          bm(1)
```

DTProbLog

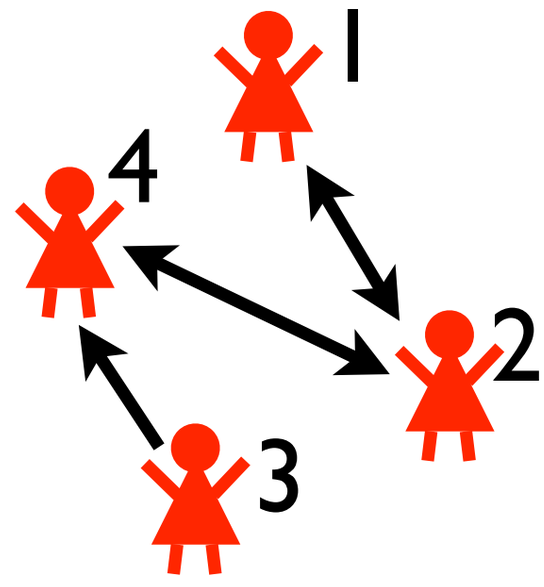



```
? :: marketed(P) :- person(P).

0.3 :: buy_trust(X,Y) :- friend(X,Y).
0.2 :: buy_marketing(P) :- person(P).

buys(X) :- friend(X,Y), buys(Y), buy_trust(X,Y).
buys(X) :- marketed(X), buy_marketing(X).

buys(P) => 5 :- person(P).
marketed(P) => -3 :- person(P).
```

```
person(1).
person(2).
person(3).
person(4).

friend(1,2).
friend(2,1).
friend(2,4).
friend(3,4).
friend(4,2).
```

```
marketed(1)        marketed(3)

    bt(2,1)   bt(2,4)        bm(1)

buys(1)    buys(2)
```

DTProbLog

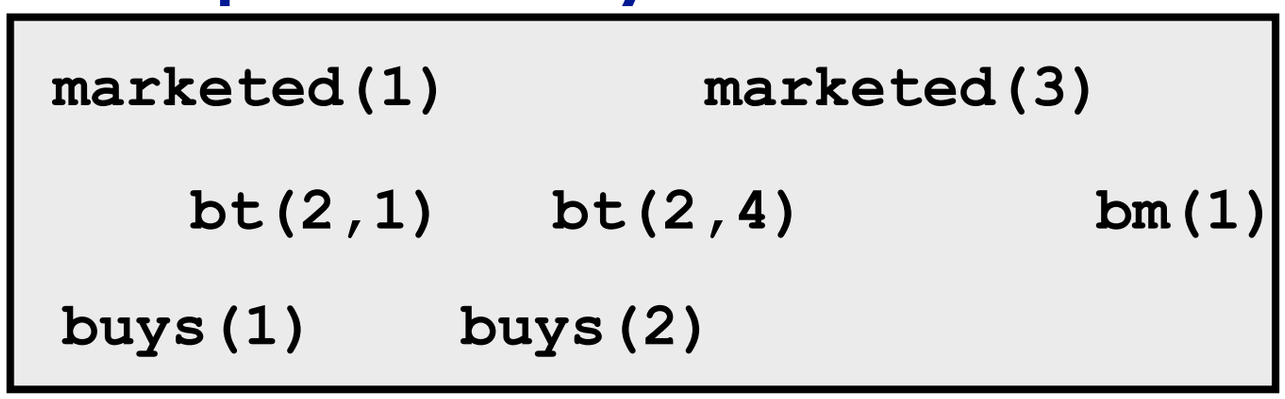



```
? :: marketed(P) :- person(P).

0.3 :: buy_trust(X,Y) :- friend(X,Y).
0.2 :: buy_marketing(P) :- person(P).

buys(X) :- friend(X,Y), buys(Y), buy_trust(X,Y).
buys(X) :- marketed(X), buy_marketing(X).

buys(P) => 5 :- person(P).
marketed(P) => -3 :- person(P).
```

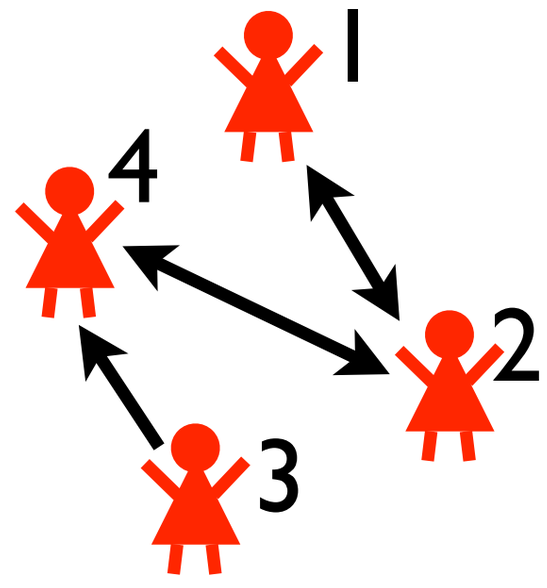
```
person(1).
person(2).
person(3).
person(4).

friend(1,2).
friend(2,1).
friend(2,4).
friend(3,4).
friend(4,2).
```

utility = −3 + −3 + 5 + 5 = 4

probability = 0.0032



DTProbLog




```
? :: marketed(P) :- person(P).

0.3 :: buy_trust(X,Y) :- friend(X,Y).
0.2 :: buy_marketing(P) :- person(P).

buys(X) :- friend(X,Y), buys(Y), buy_trust(X,Y).
buys(X) :- marketed(X), buy_marketing(X).

buys(P) => 5 :- person(P).
marketed(P) => -3 :- person(P).
```

```
person(1).
person(2).
person(3).
person(4).

friend(1,2).
friend(2,1).
friend(2,4).
friend(3,4).
friend(4,2).
```

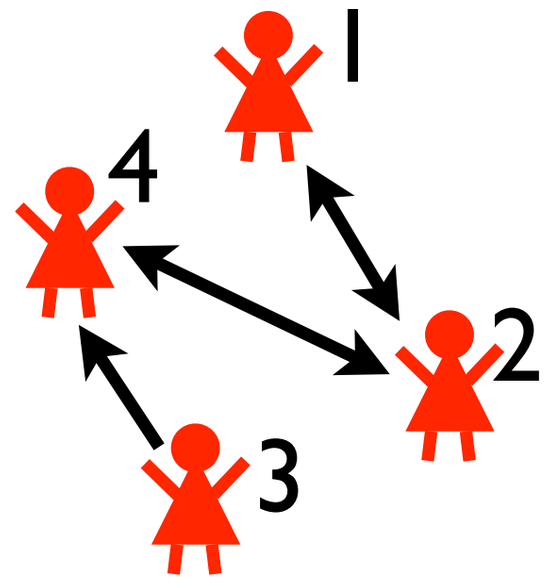
utility = −3 + −3 + 5 + 5 = 4

probability = 0.0032

```
marketed(1)        marketed(3)

   bt(2,1)   bt(2,4)          bm(1)

buys(1)    buys(2)
```

world contributes 0.0032×4 to expected utility of strategy

DTProbLog




```
? :: marketed(P) :- person(P).

0.3 :: buy_trust(X,Y) :- friend(X,Y).
0.2 :: buy_marketing(P) :- person(P).

buys(X) :- friend(X,Y), buys(Y), buy_trust(X,Y).
buys(X) :- marketed(X), buy_marketing(X).

buys(P) => 5 :- person(P).
marketed(P) => -3 :- person(P).
```

```
person(1).
person(2).
person(3).
person(4).

friend(1,2).
friend(2,1).
friend(2,4).
friend(3,4).
friend(4,2).
```

task: find strategy that maximizes expected utility
solution: using ProbLog technology

Phenetic

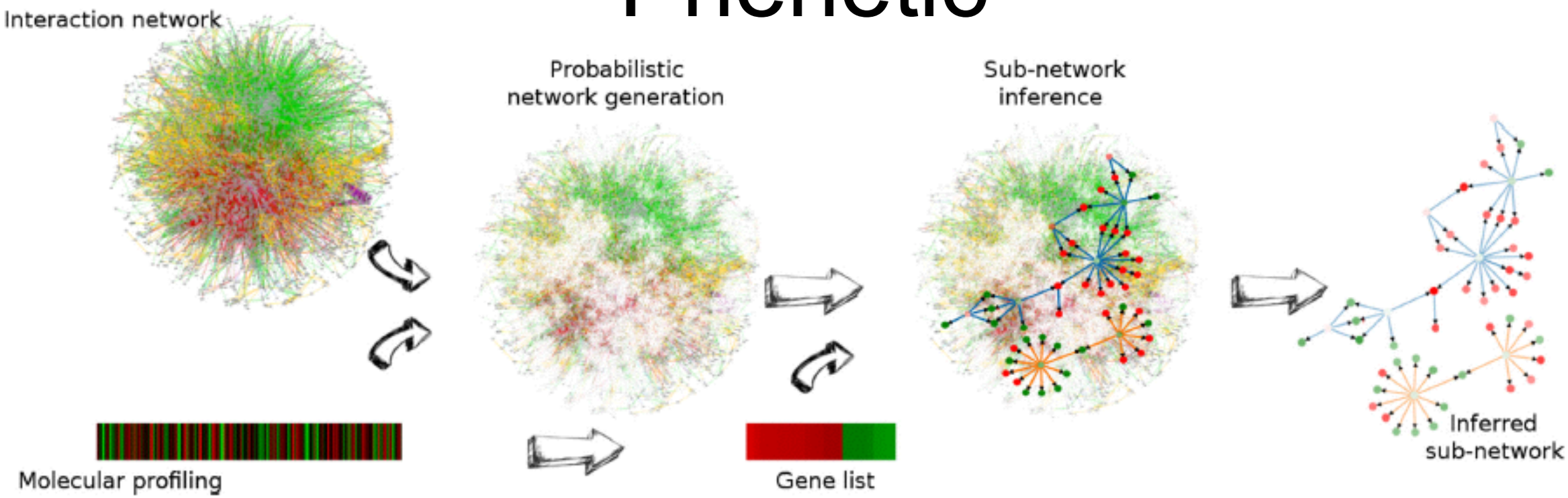


Figure 1. Overview of PheNetic, a web service for network-based interpretation of 'omics' data. The web service uses as input a genome wide interaction network for the organism of interest, a user generated molecular profiling data set and a gene list derived from these data. Interaction networks for a wide variety of organisms are readily available from the web server. Using the uploaded user-generated molecular data the interaction network is converted into a probabilistic network: edges receive a probability proportional to the levels measured for the terminal nodes in the molecular profiling data set. This probabilistic interaction network is used to infer the sub-network that best links the genes from the gene list. The inferred sub-network provides a trade-off between linking as many genes as possible from the gene list and selecting the least number of edges.

- Causes: Mutations
  - All related to similar phenotype
- Effects: Differentially expressed genes
- 27 000 cause effect pairs

- Interaction network:
  - 3063 nodes
    - Genes
    - Proteins
  - 16794 edges
    - Molecular interactions
    - Uncertain

- Goal: connect causes to effects through common subnetwork
  - = Find mechanism
- Techniques:
  - DTProbLog
  - Approximate inference

[De Maeyer et al., Molecular Biosystems 13, NAR 15]

Roadmap

- Modeling
- Inference
- Learning & Dynamics
- Advanced Inference and KBMC

... with some detours on the way

Part V: KBMC

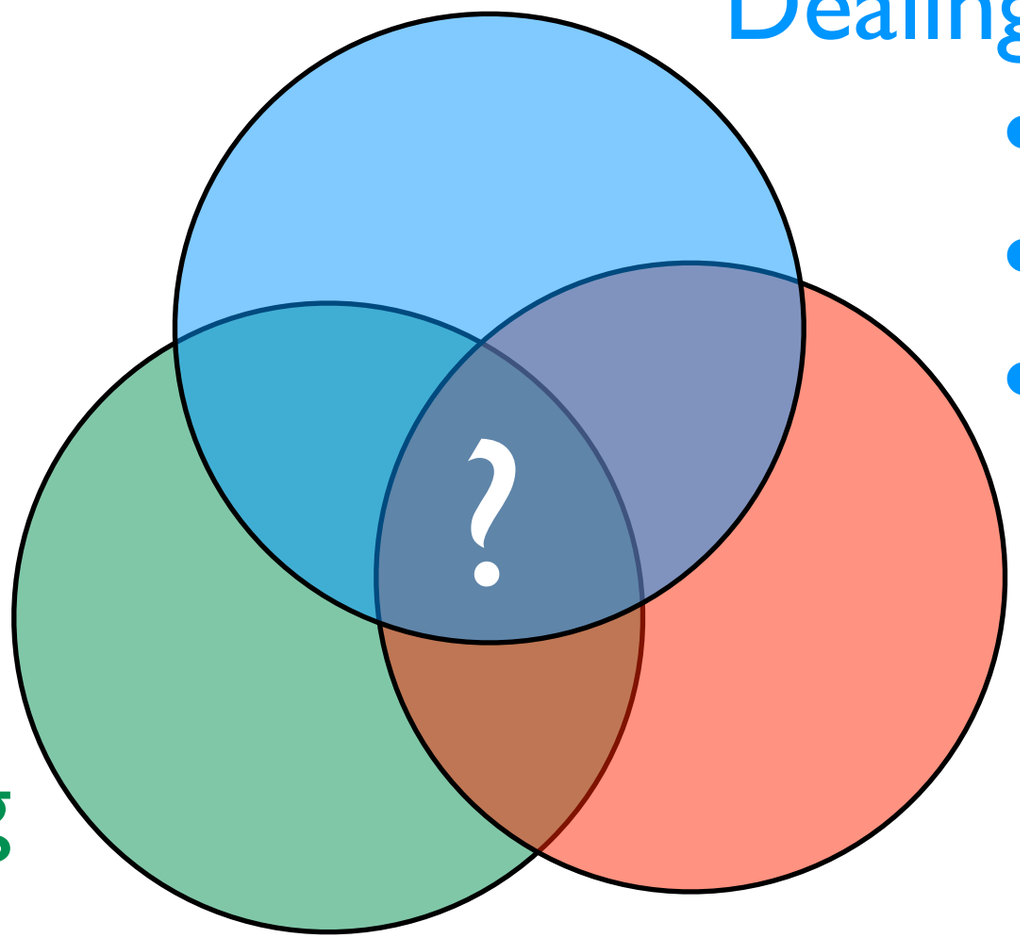
A key question in AI:

Dealing with uncertainty

- probability theory
- graphical models
- ...

Reasoning with relational data

- logic
- databases
- programming
- ...



Learning

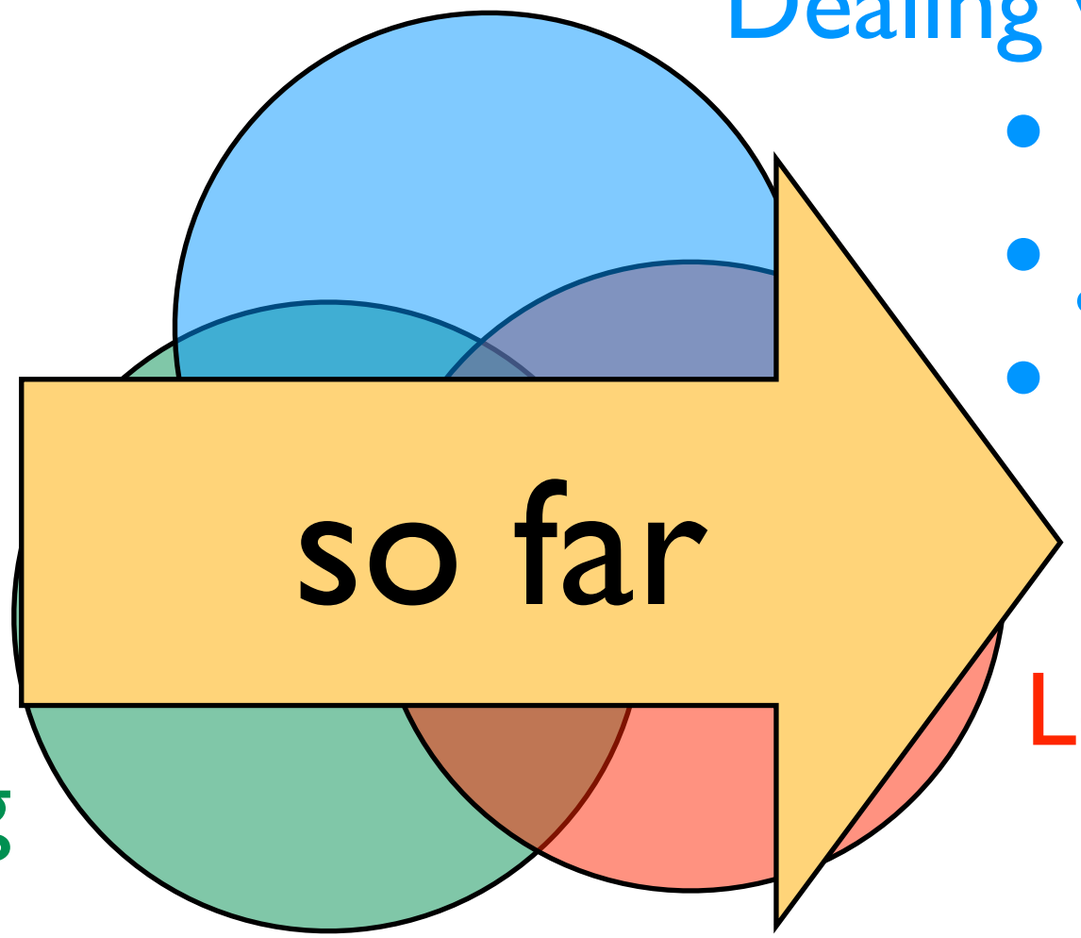
- parameters
- structure

Statistical relational learning
& Probabilistic programming, ...

A key question in AI:

Reasoning with relational data

- logic
- databases
- programming
- ...



Dealing with uncertainty

- probability theory
- graphical models
- ...

Learning

- parameters
- structure

Statistical relational learning
& Probabilistic programming, ...

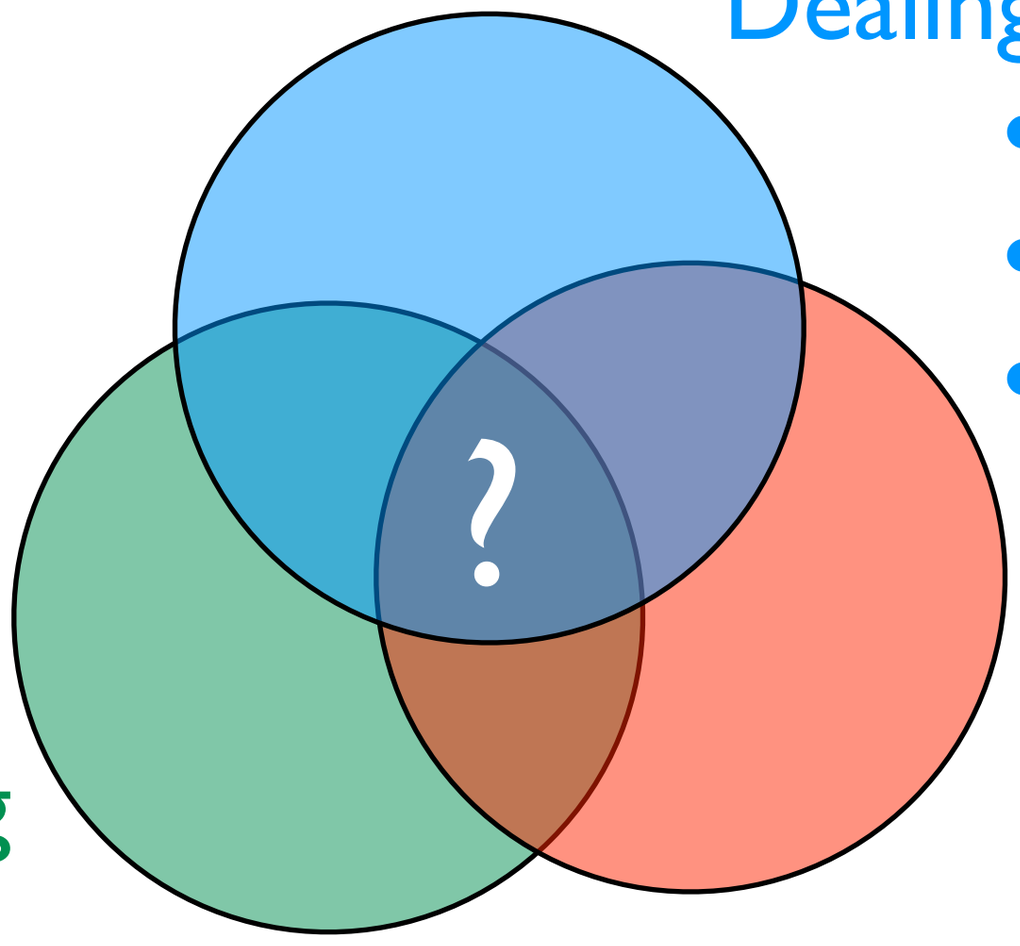
A key question in AI:

Dealing with uncertainty

- probability theory
- graphical models
- ...

Reasoning with relational data

- logic
- databases
- programming
- ...



Learning

- parameters
- structure

Statistical relational learning
& Probabilistic programming, ...

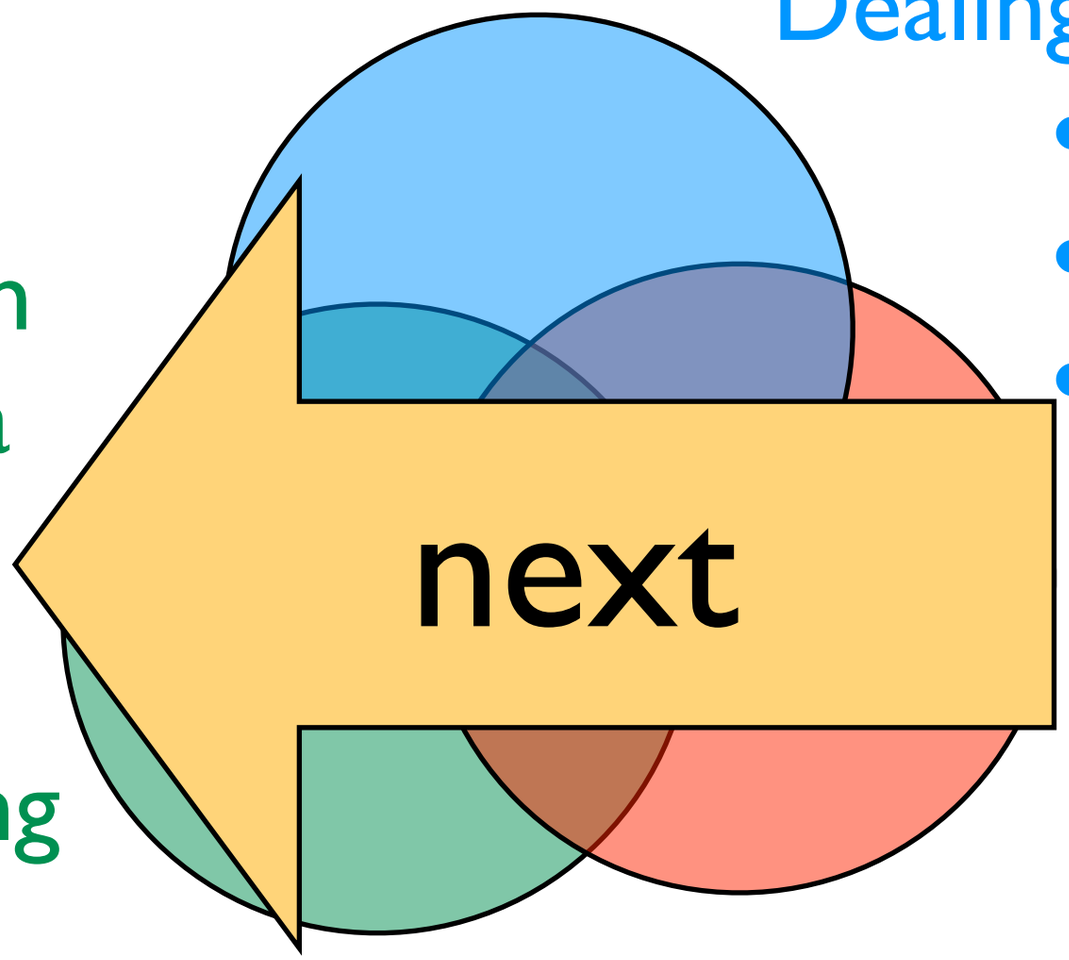
A key question in AI:

Dealing with uncertainty

- probability theory
- graphical models
- ...

Reasoning with relational data

- logic
- databases
- programming
- ...



Learning

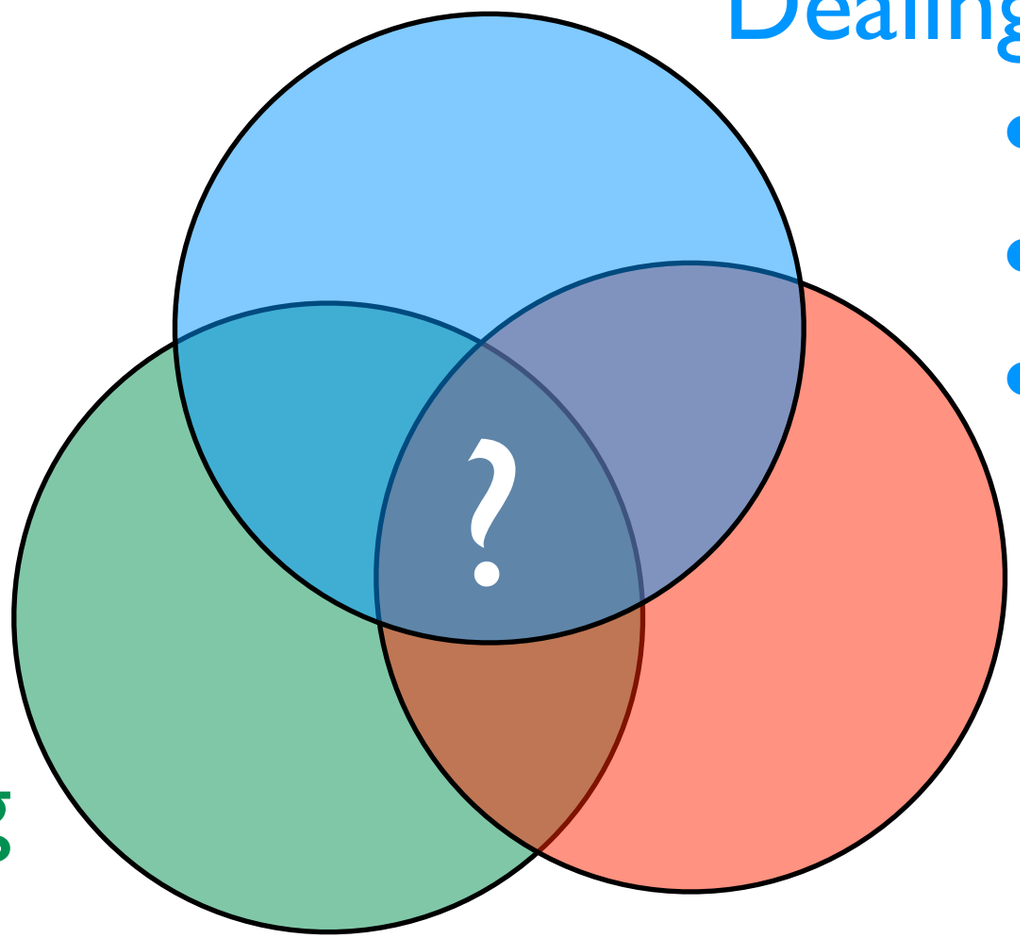
- parameters
- structure

Statistical relational learning
& Probabilistic programming, ...

A key question in AI:

Reasoning with relational data

- logic
- databases
- programming
- ...



Dealing with uncertainty

- probability theory
- graphical models
- ...

Learning

- parameters
- structure

Statistical relational learning, probabilistic logic learning, probabilistic programming, ...

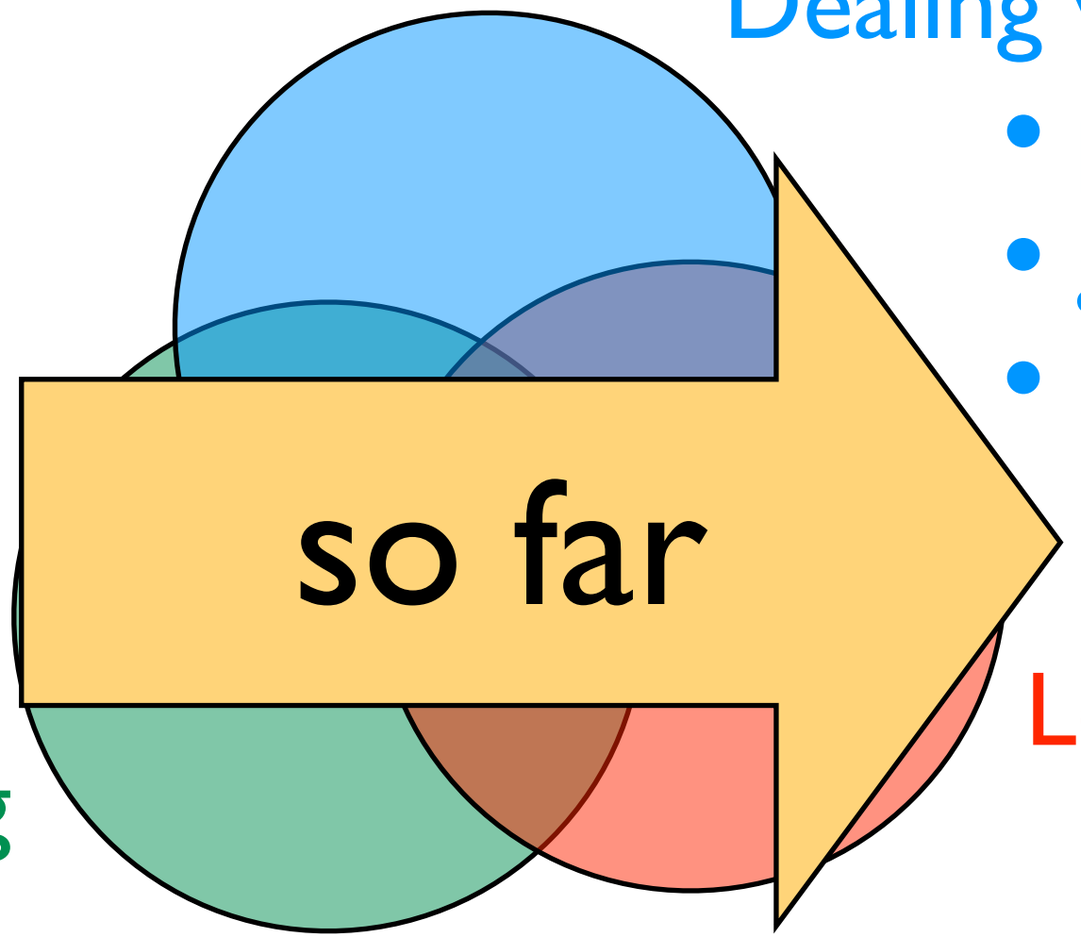
A key question in AI:

Dealing with uncertainty

- probability theory
- graphical models
- ...

Reasoning with relational data

- logic
- databases
- programming
- ...



Learning

- parameters
- structure

Statistical relational learning, probabilistic logic learning, probabilistic programming, ...

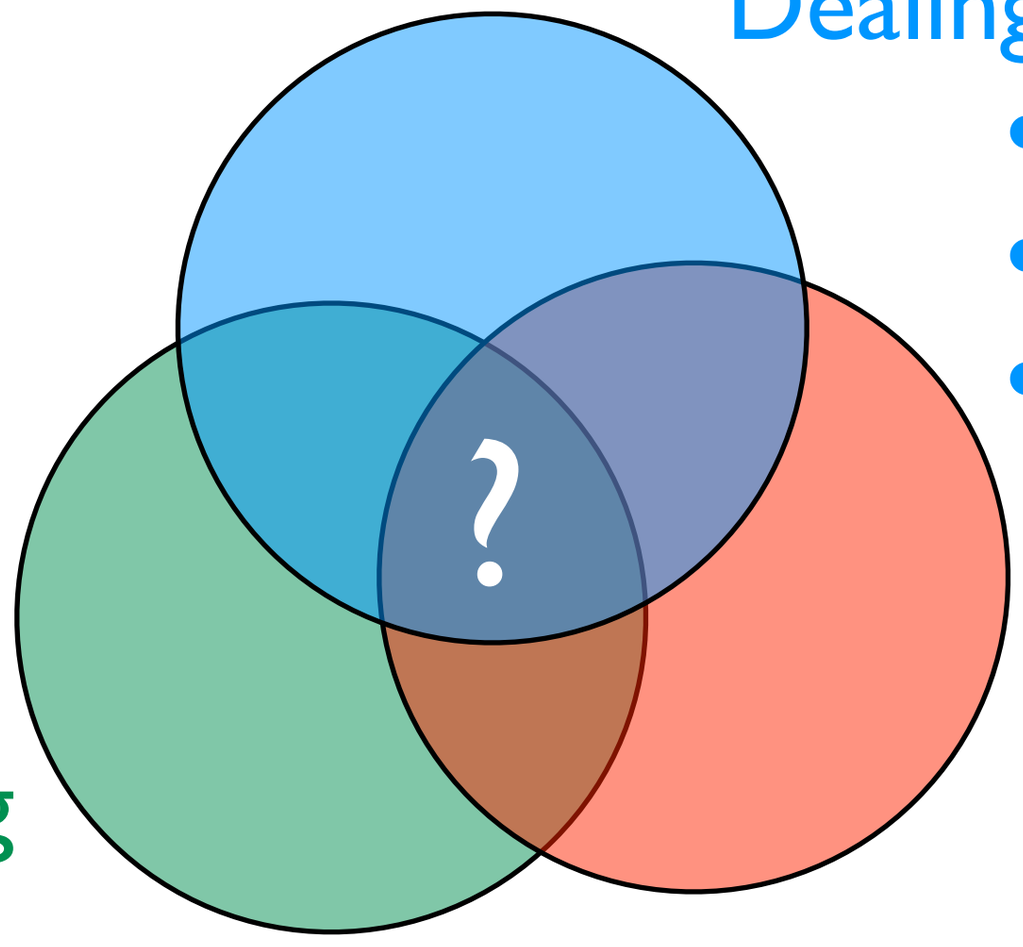
A key question in AI:

Dealing with uncertainty

- probability theory
- graphical models
- ...

Reasoning with relational data

- logic
- databases
- programming
- ...



Learning

- parameters
- structure

Statistical relational learning, probabilistic logic learning, probabilistic programming, ...

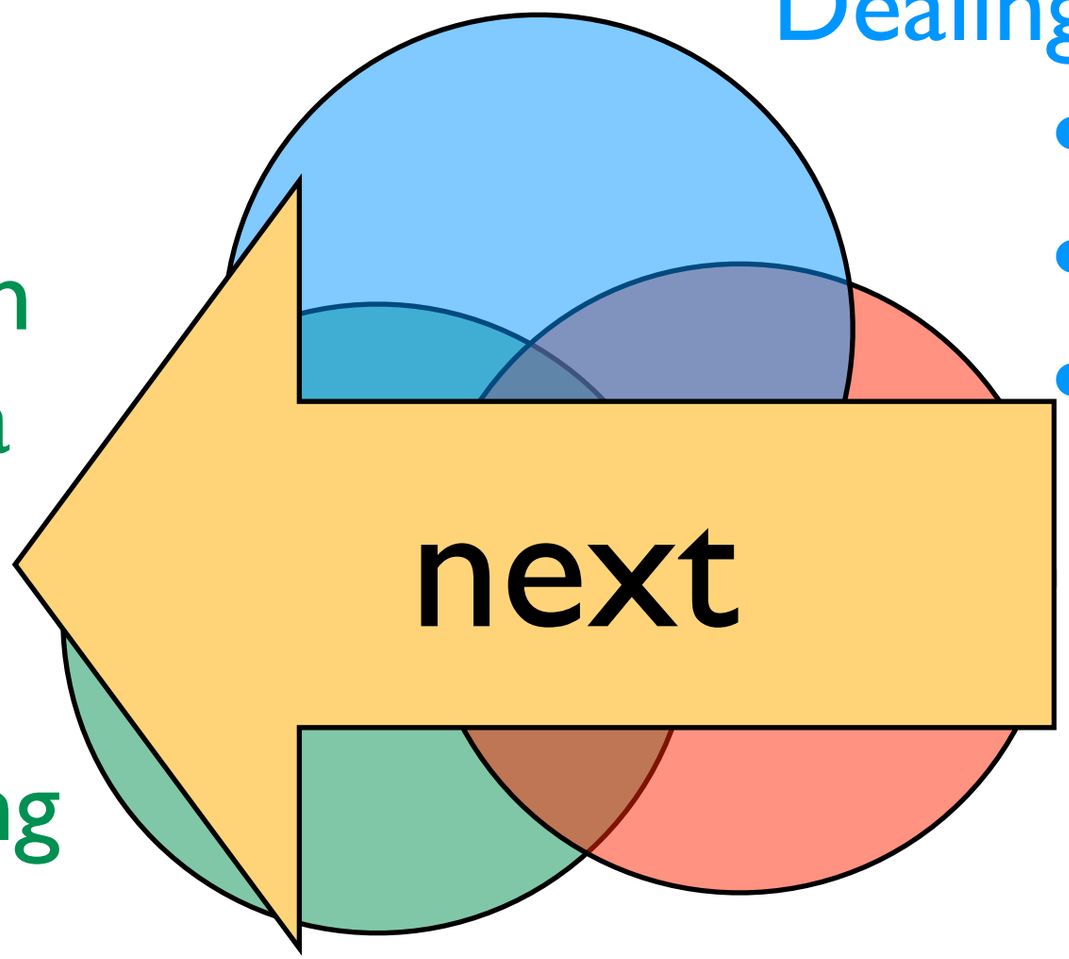
A key question in AI:

Dealing with uncertainty

- probability theory
- graphical models
- ...

Reasoning with relational data

- logic
- databases
- programming
- ...



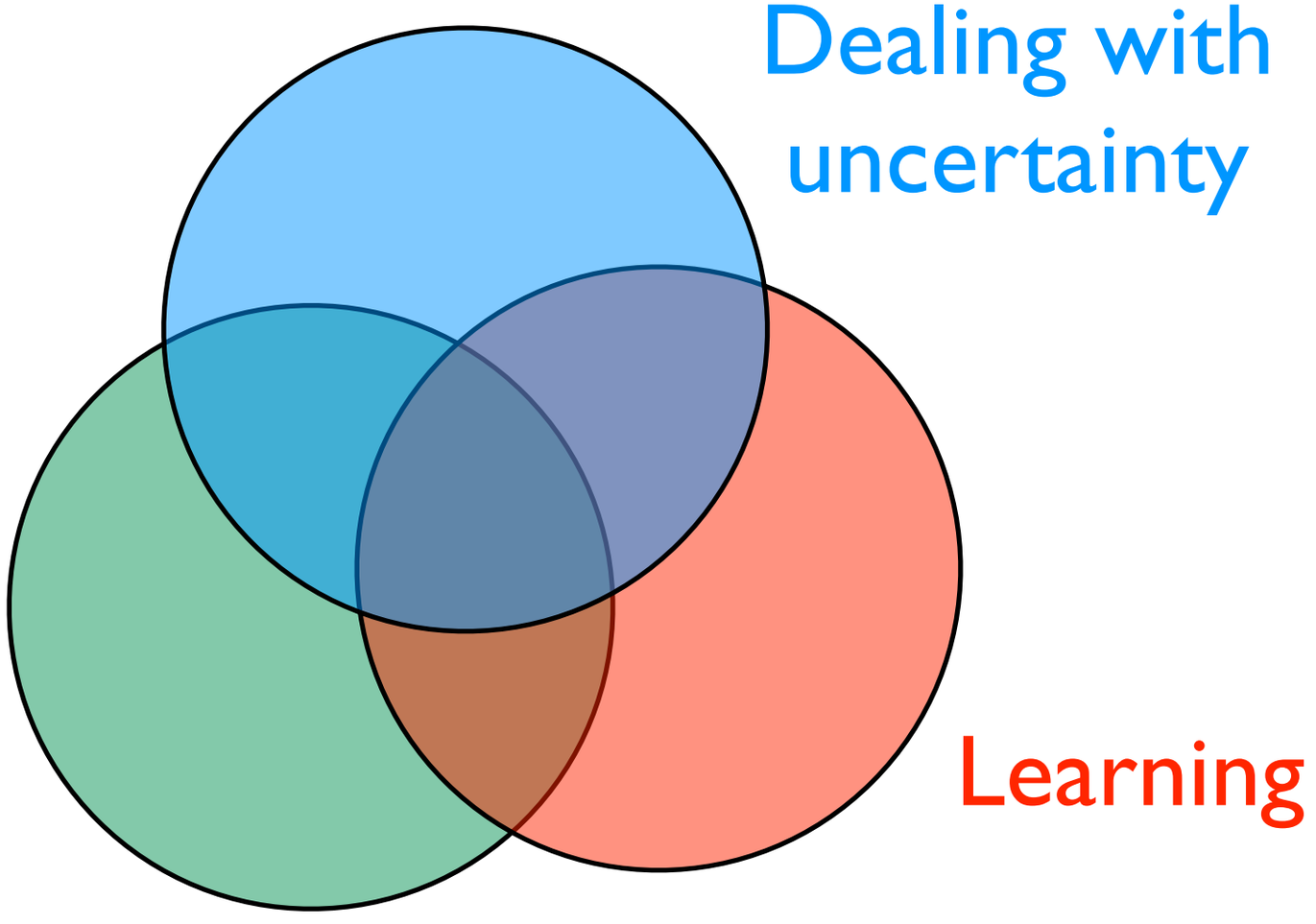
Learning

- parameters
- structure

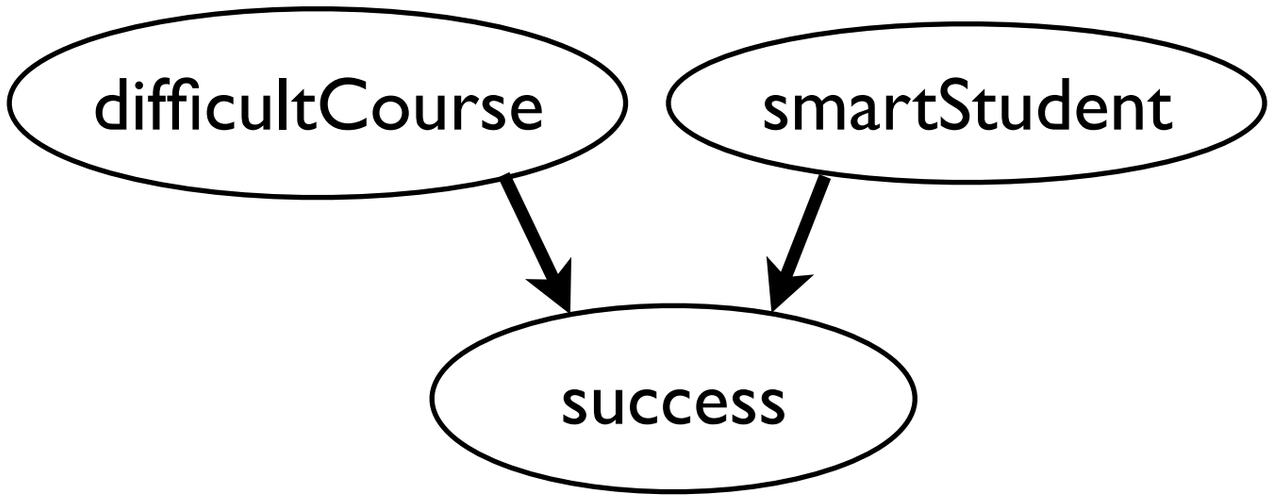
Statistical relational learning, probabilistic logic learning, probabilistic programming, ...

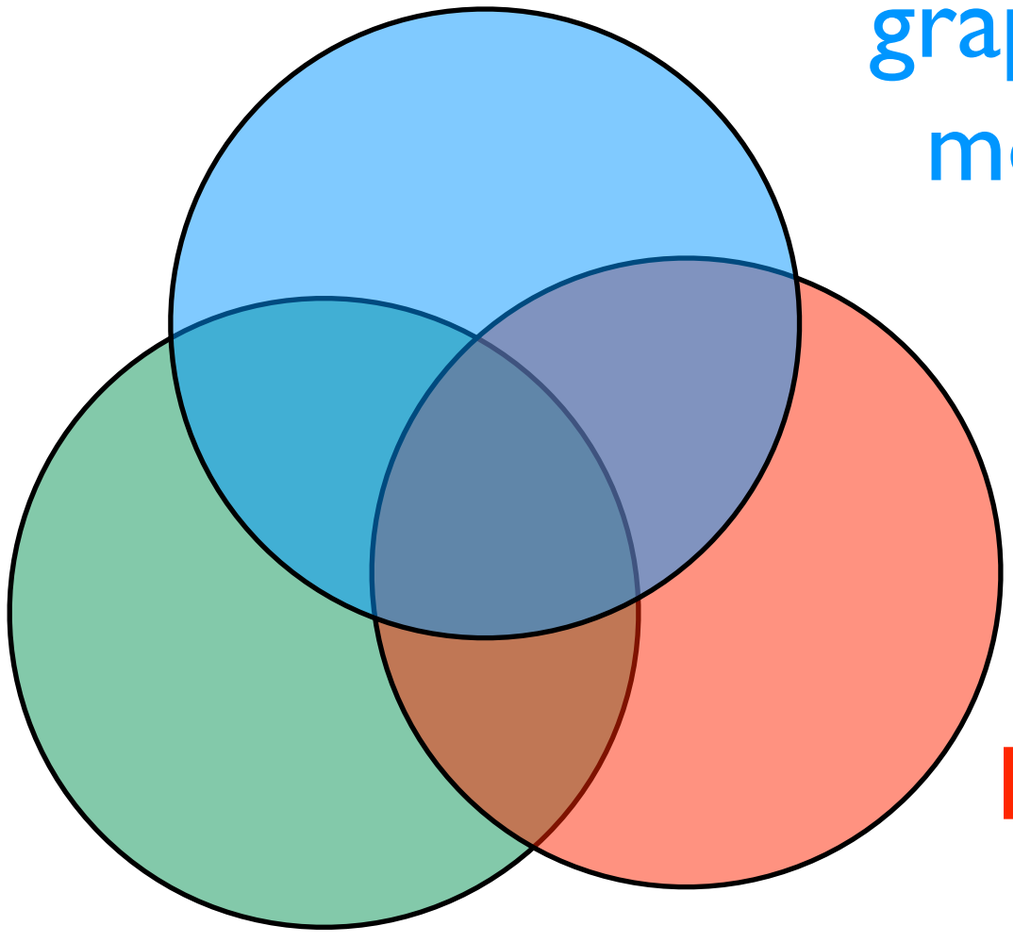
lifted graphical models

Lifted graphical models



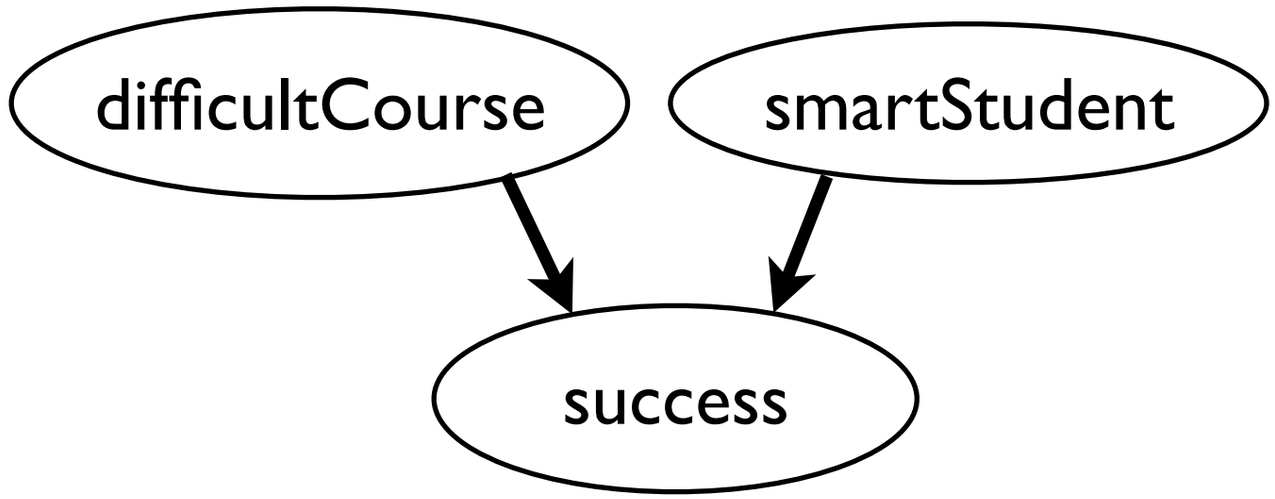
Lifted graphical models

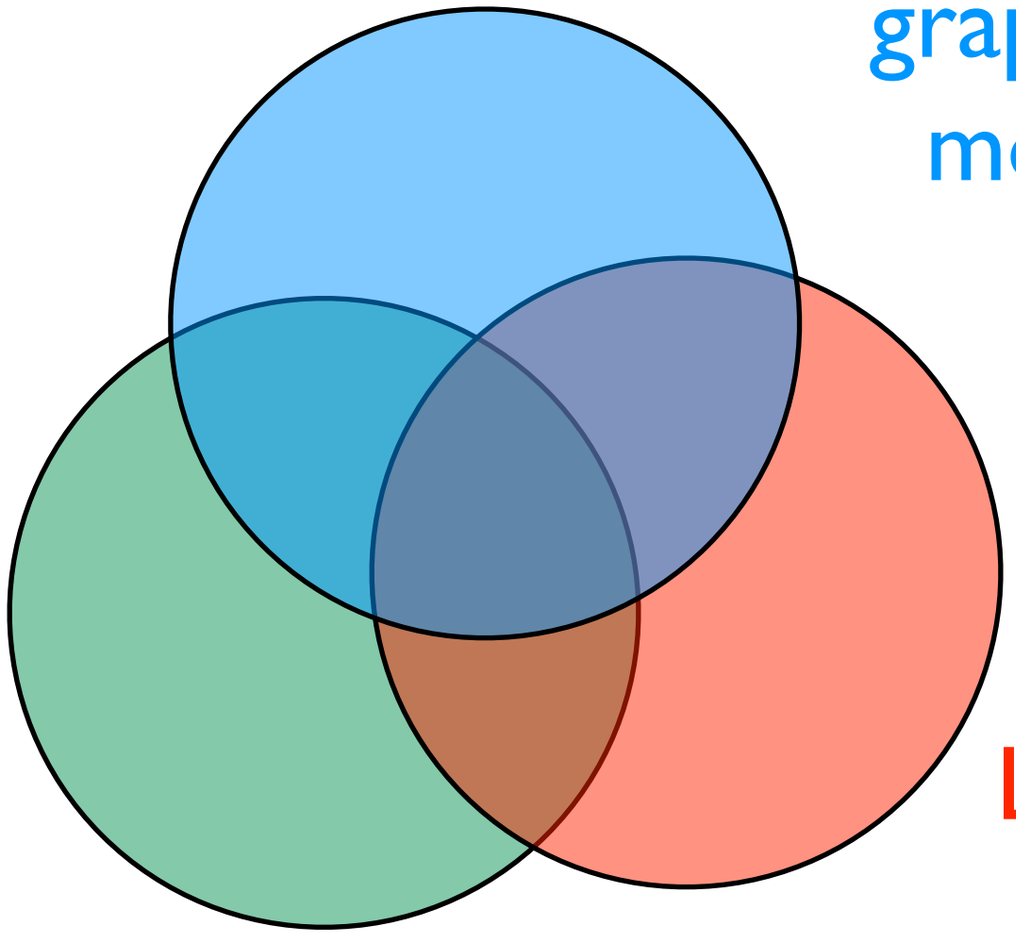




Lifted graphical models

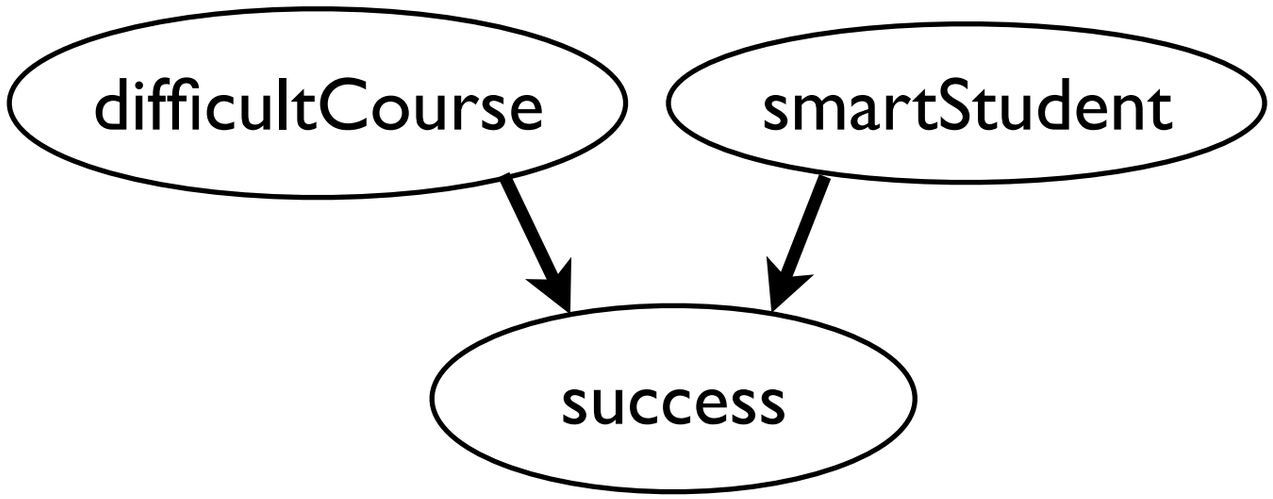
fixed set of random variables



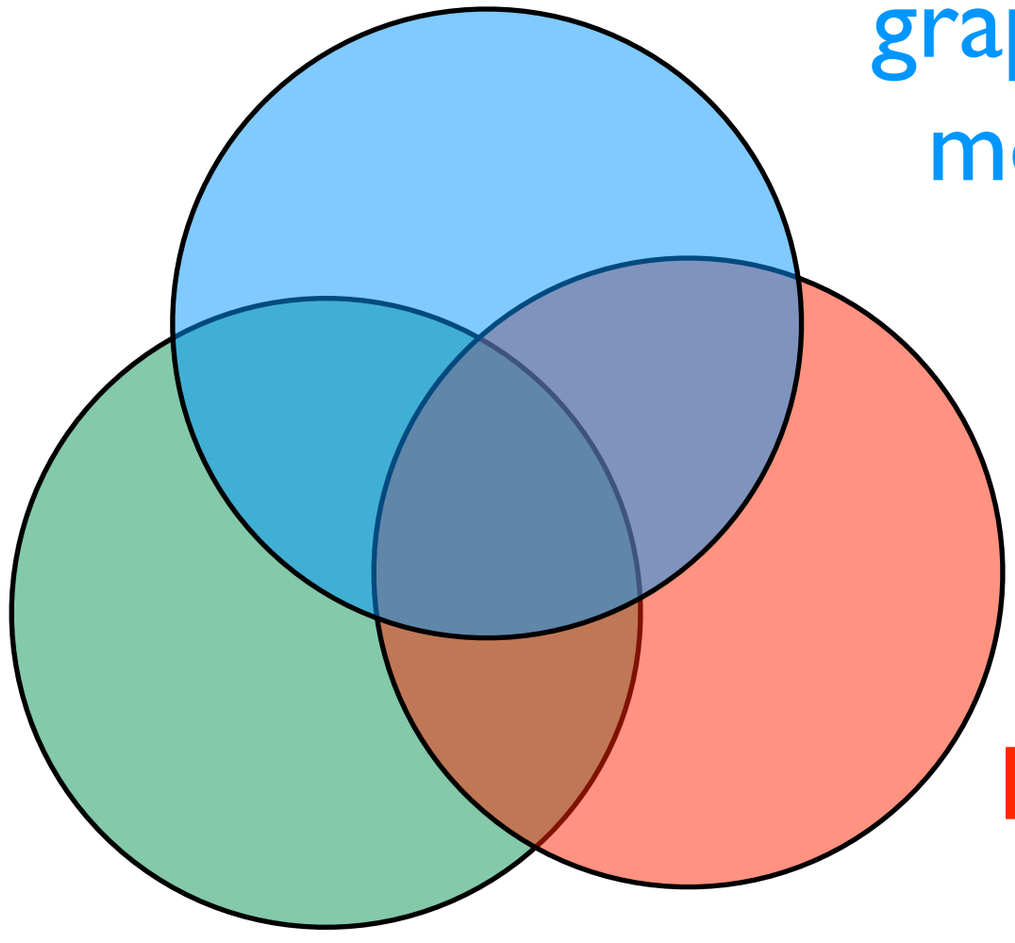


Lifted graphical models

fixed set of random variables

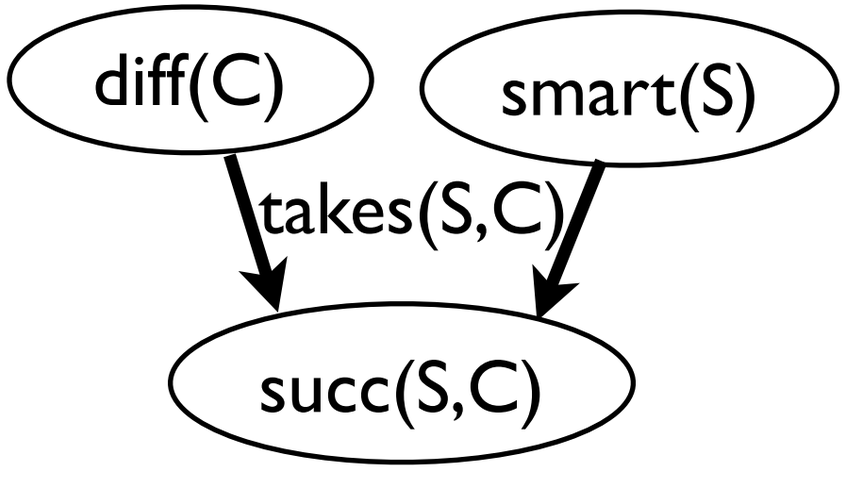


graphical model



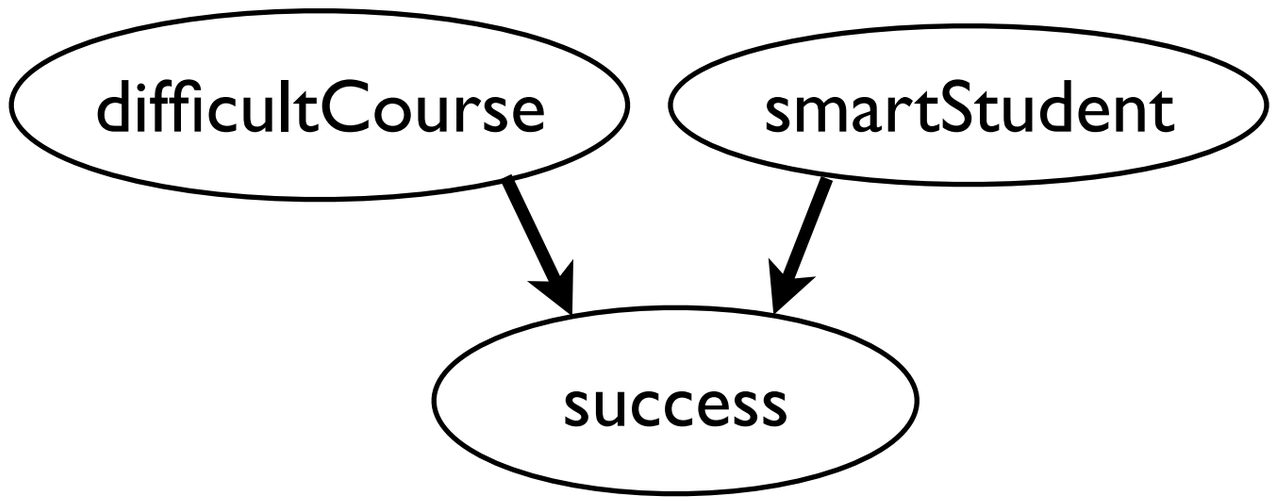
Learning

relational definition of graphical model

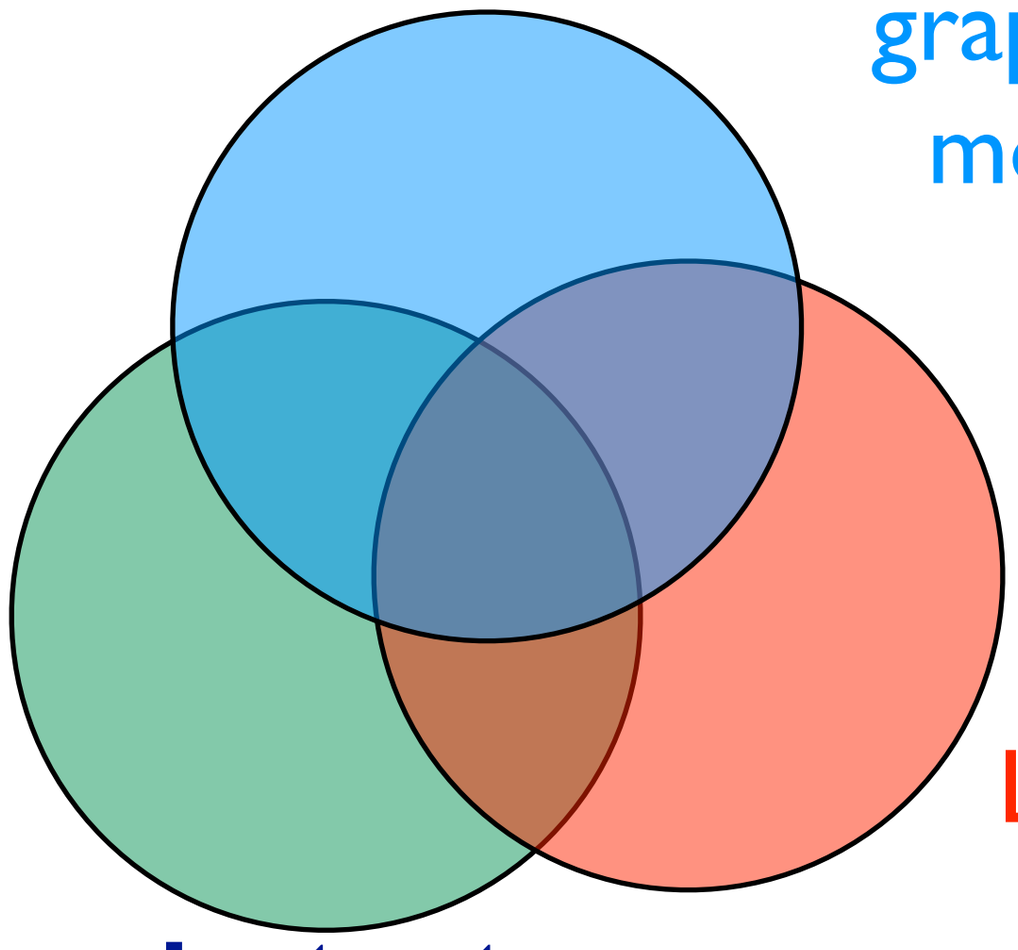


Lifted graphical models

fixed set of random variables

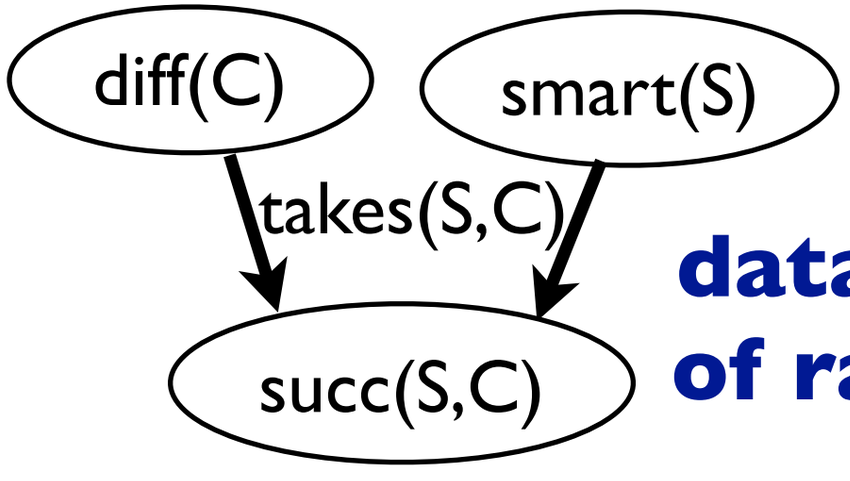


graphical model



Learning

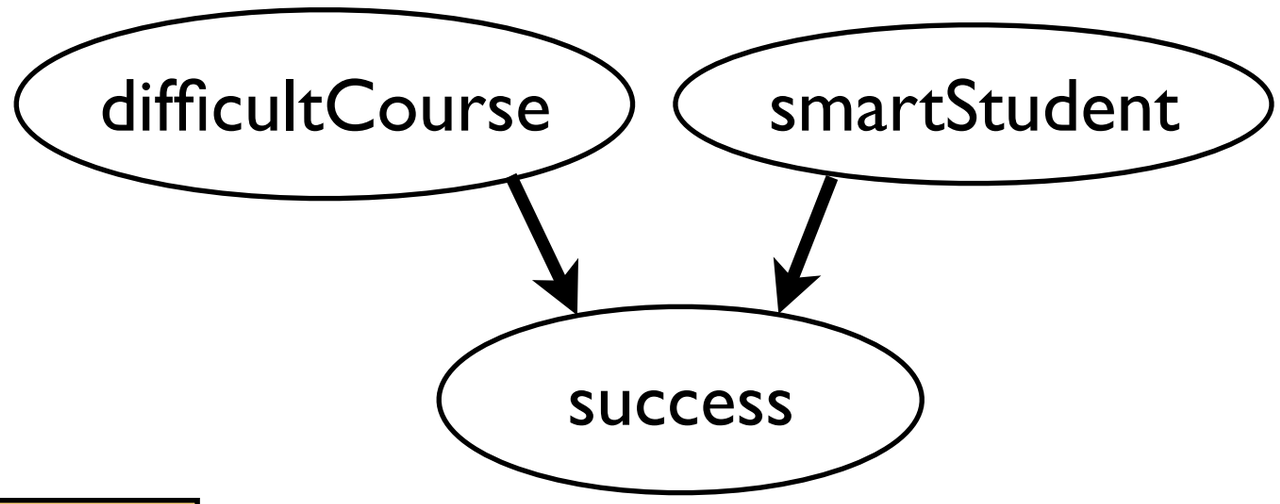
relational definition of graphical model



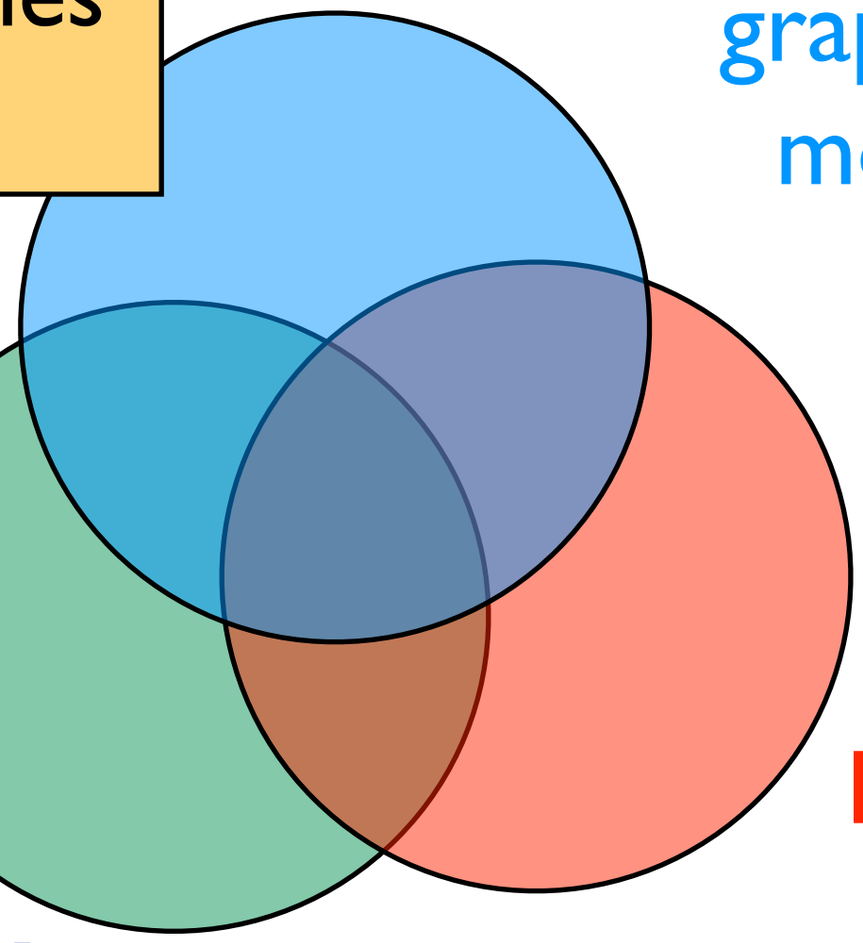
data-dependent set of random variables

Lifted graphical models

fixed set of random variables



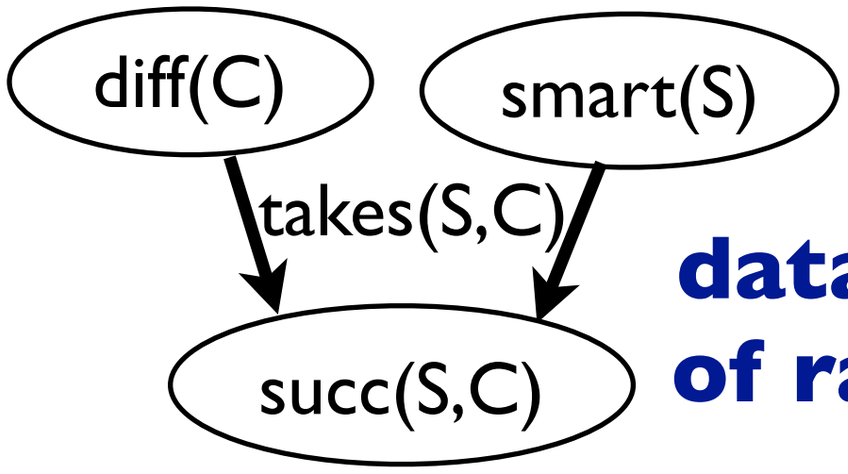
relational language defines graphical model



graphical model

relational definition of graphical model

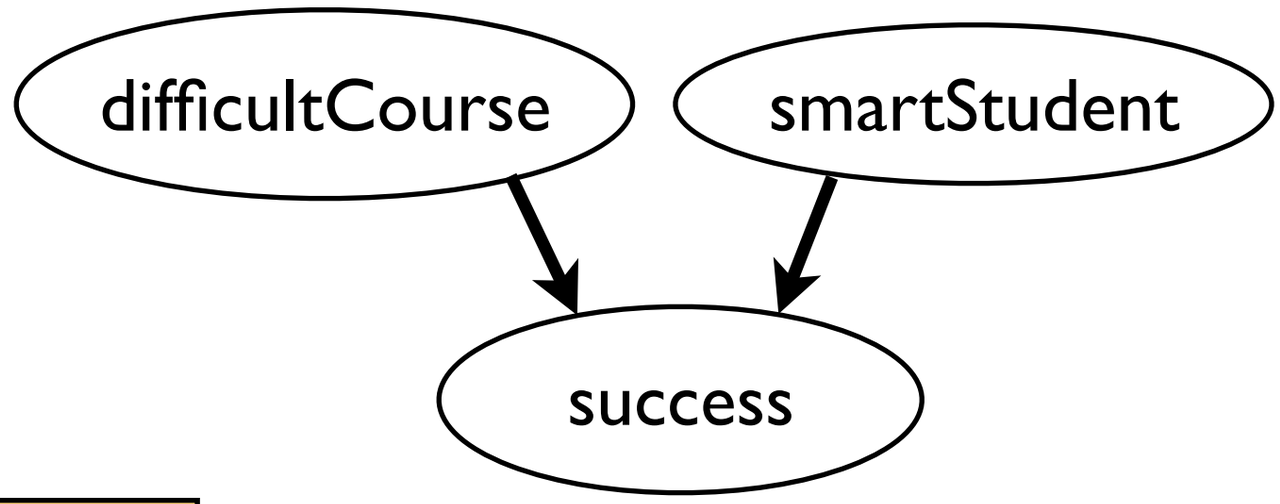
Learning



data-dependent set of random variables

Lifted graphical models

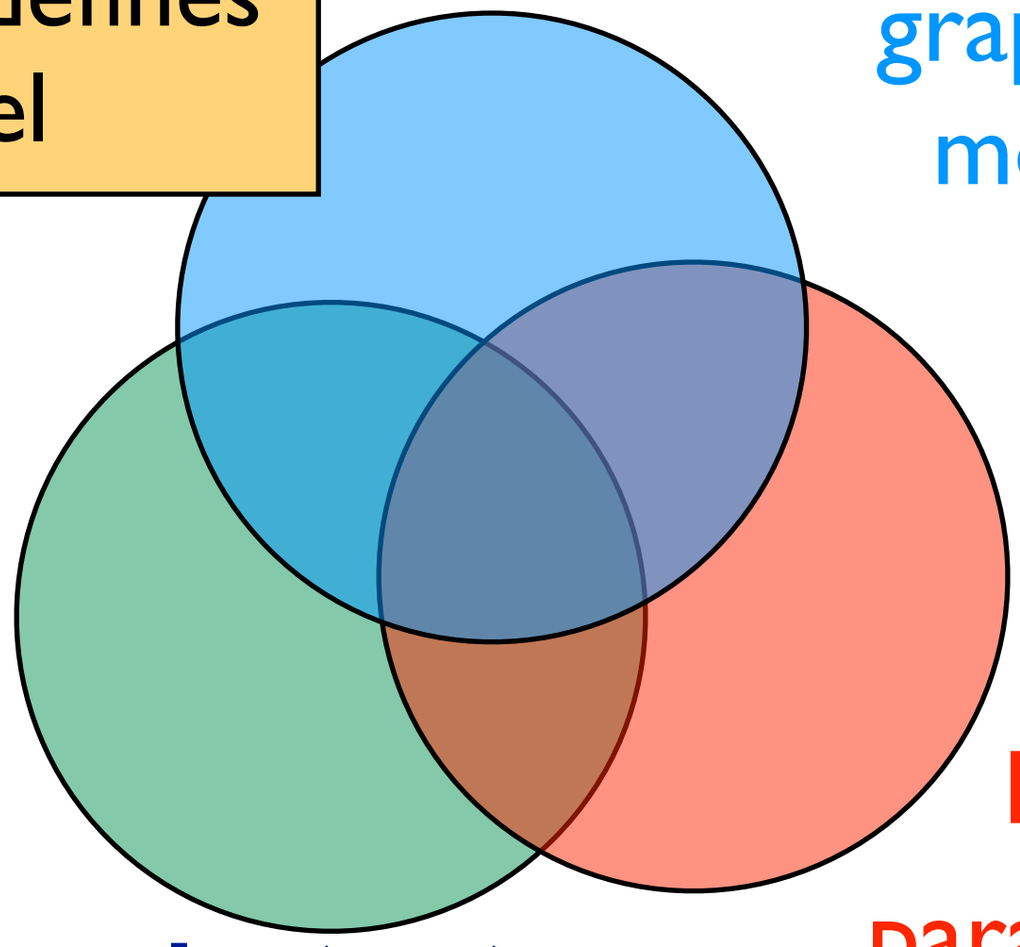
fixed set of random variables



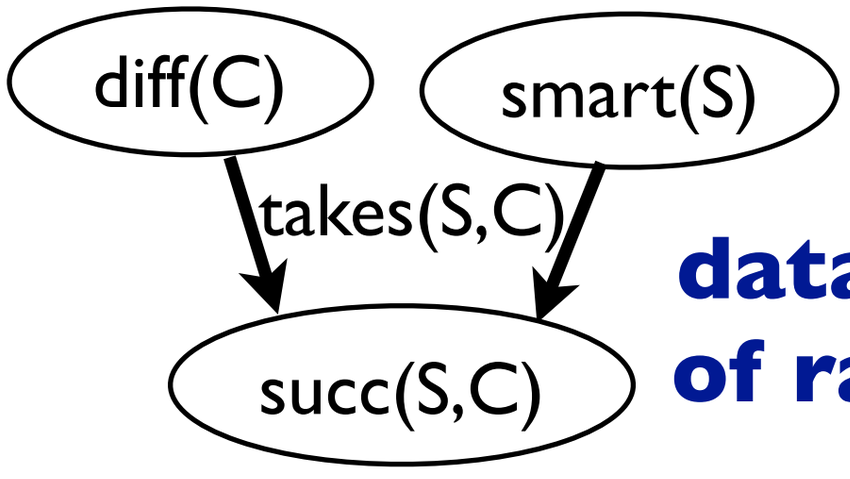
relational language defines graphical model

graphical model

relational definition of graphical model



Learning parameters & structure

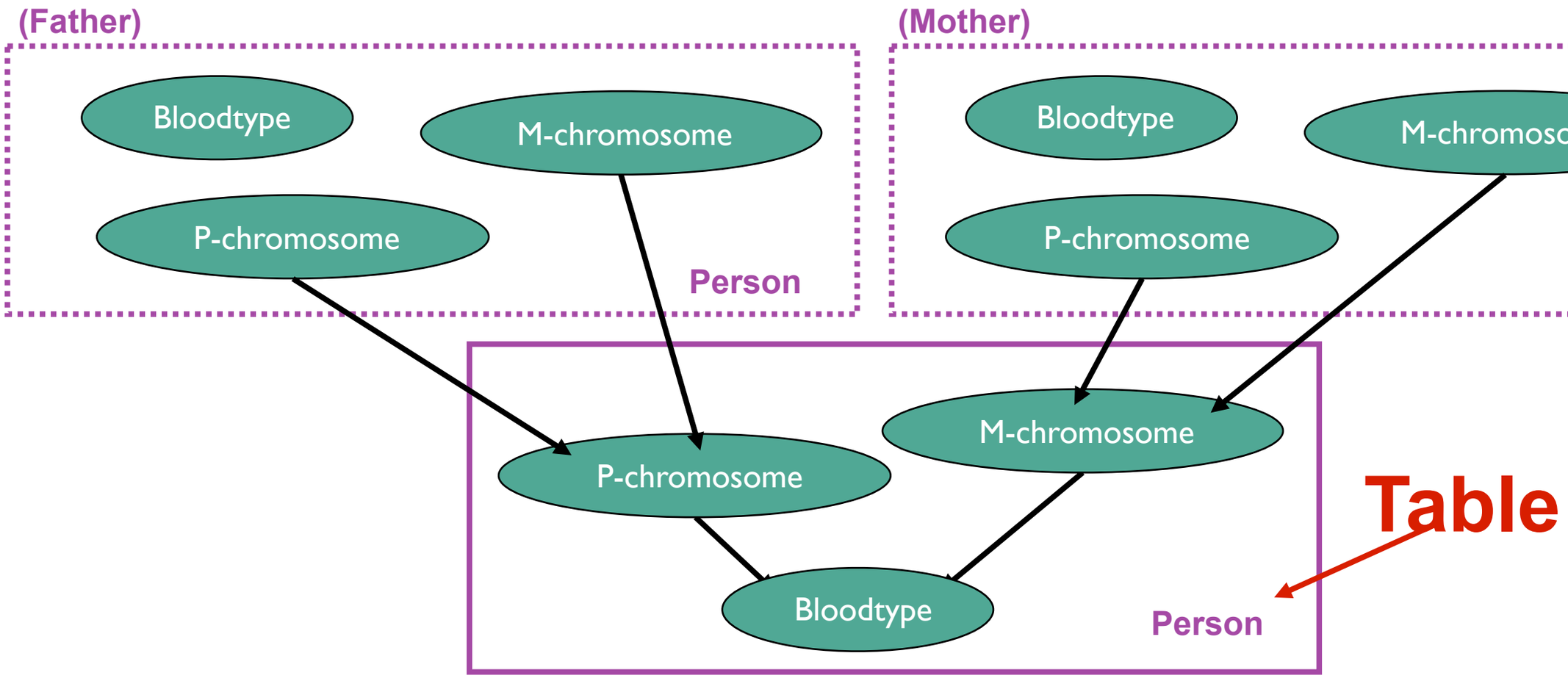


data-dependent set of random variables

Lots of proposals in the literature, e.g.

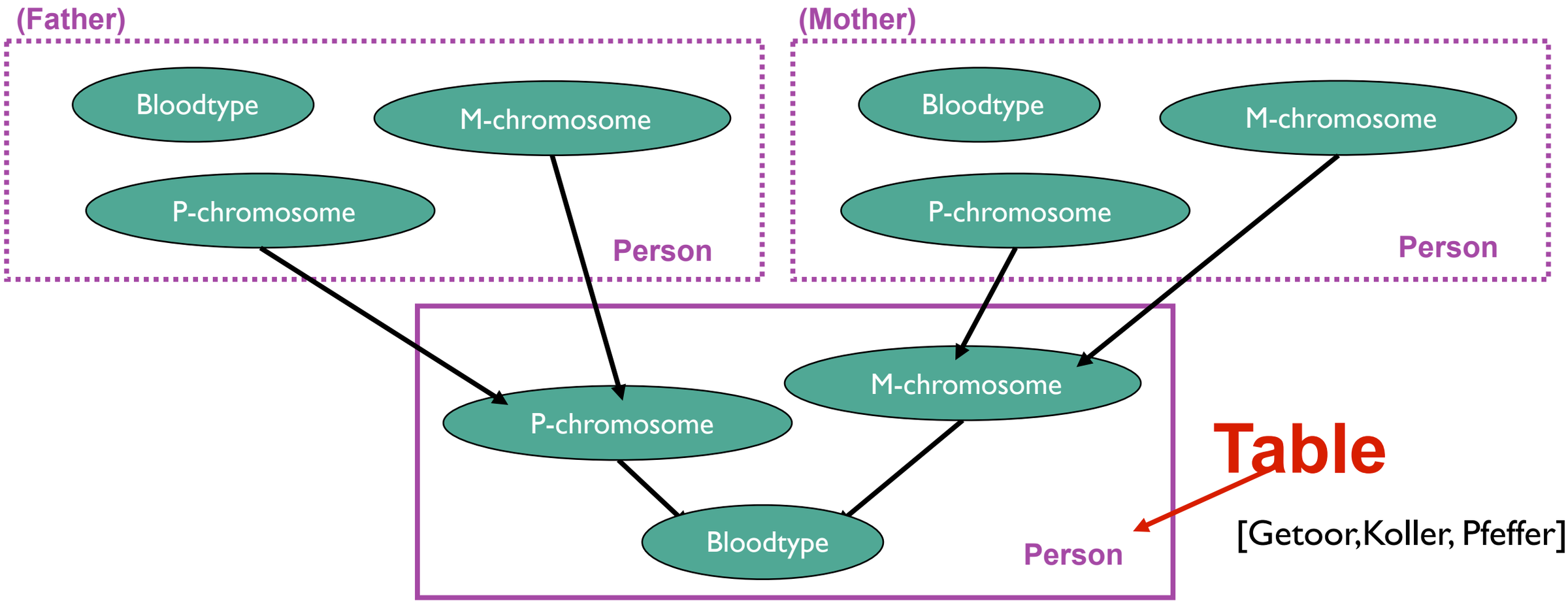
- relational Markov networks (RMNs) [Taskar et al 2002]
- **Markov logic networks (MLNs)** [Richardson & Domingos 2006]
- *probabilistic soft logic (PSL) [Broecheler et al 2010]*
- FACTORIE [McCallum et al 2009]
- Bayesian logic programs (BLPs) [Kersting & De Raedt 2001]
- relational Bayesian networks (RBNs) [Jaeger 2002]
- logical Bayesian networks (LBNs) [Fierens et al 2005]
- probabilistic relational models (PRMs) [Koller & Pfeffer 1998]
- Bayesian logic (BLOG) [Milch et al 2005]
- CLP(BN) [Santos Costa et al 2008]
- and many more ...

Probabilistic Relational Models (PRMs)



[Getoor,Koller, Pfeffer]

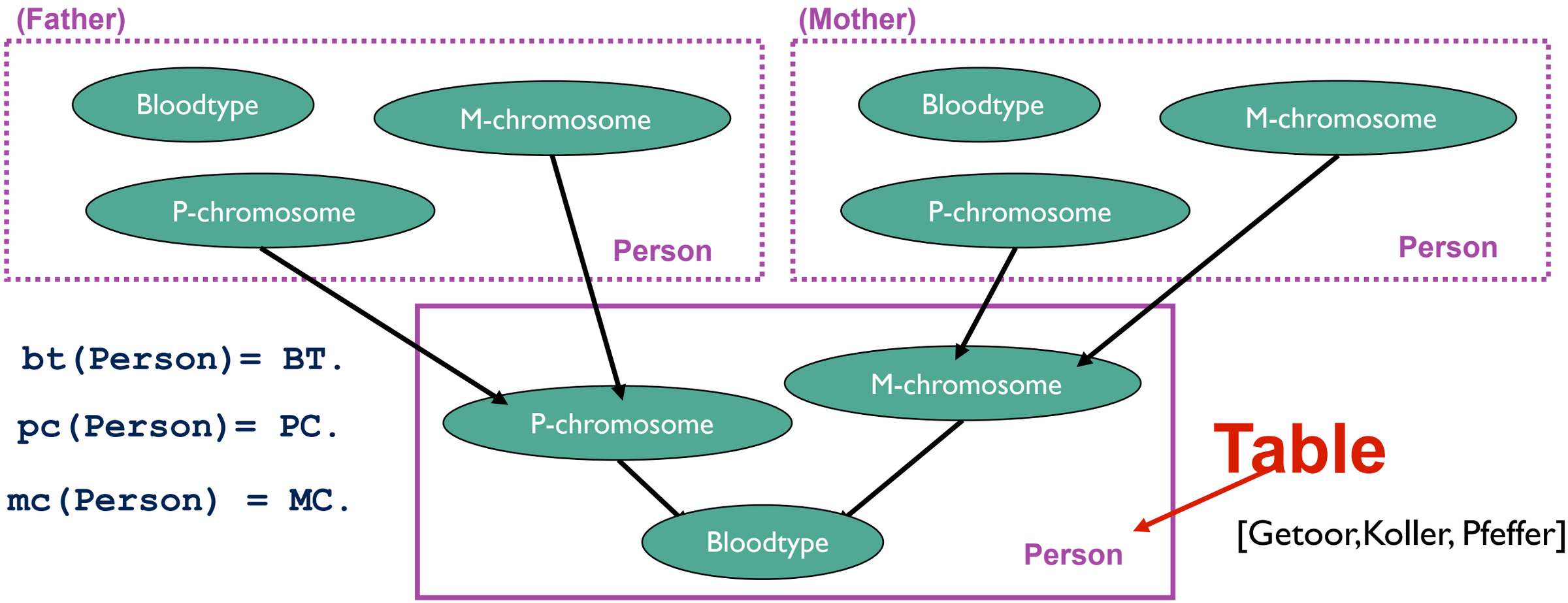
Probabilistic Relational Models (PRMs)



Table

[Getoor,Koller, Pfeffer]

Probabilistic Relational Models (PRMs)

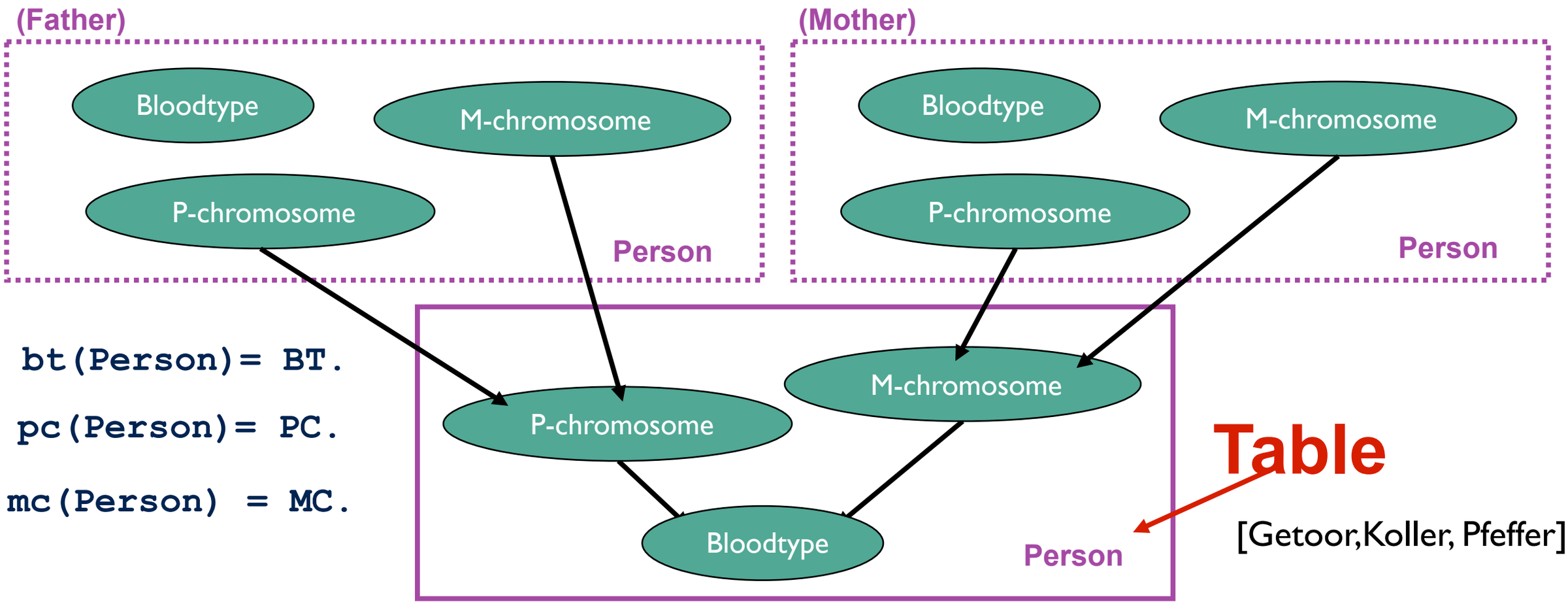



```
bt(Person)= BT.
pc(Person)= PC.
mc(Person) = MC.
```

Table

[Getoor,Koller, Pfeffer]

Probabilistic Relational Models (PRMs)




```
bt(Person)= BT.
pc(Person)= PC.
mc(Person) = MC.
```

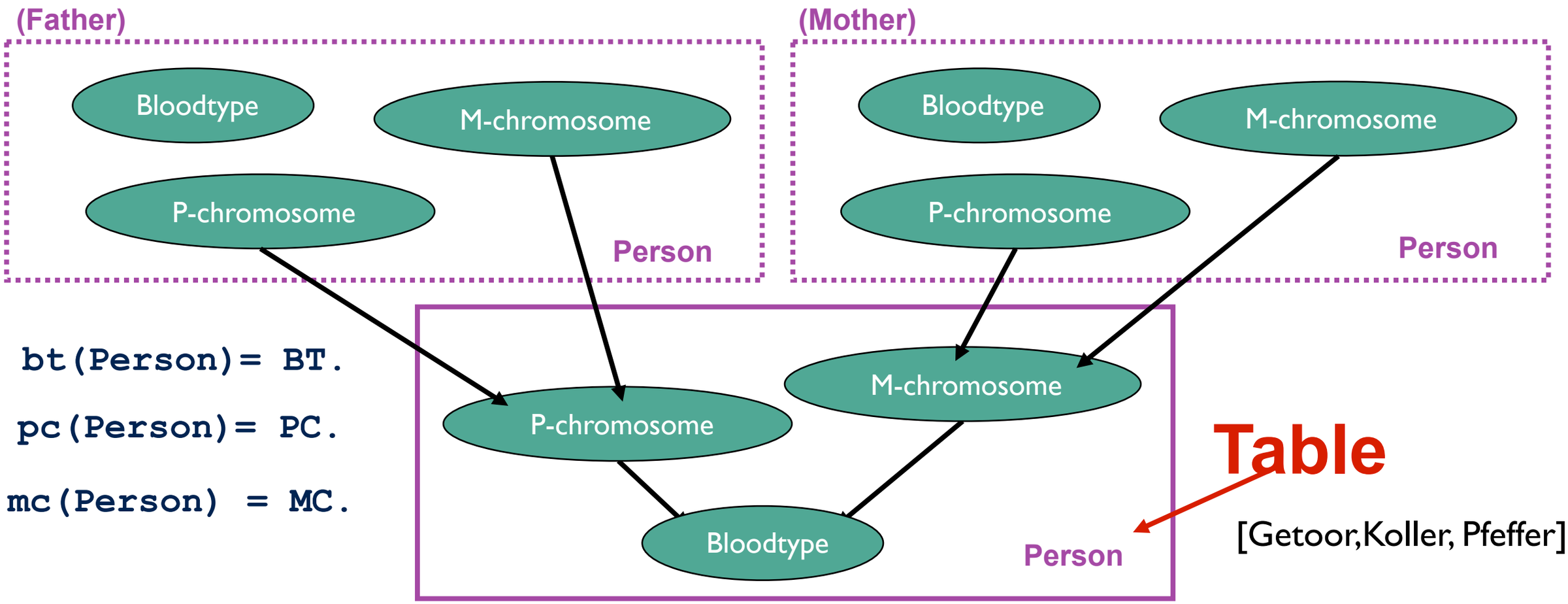
Table

[Getoor,Koller, Pfeffer]

Dependencies (CPDs associated with):

```
bt(Person)=BT | pc(Person) =PC , mc(Person) =MC.
```

Probabilistic Relational Models (PRMs)




```
bt(Person)= BT.
pc(Person)= PC.
mc(Person) = MC.
```

Table

[Getoor,Koller, Pfeffer]

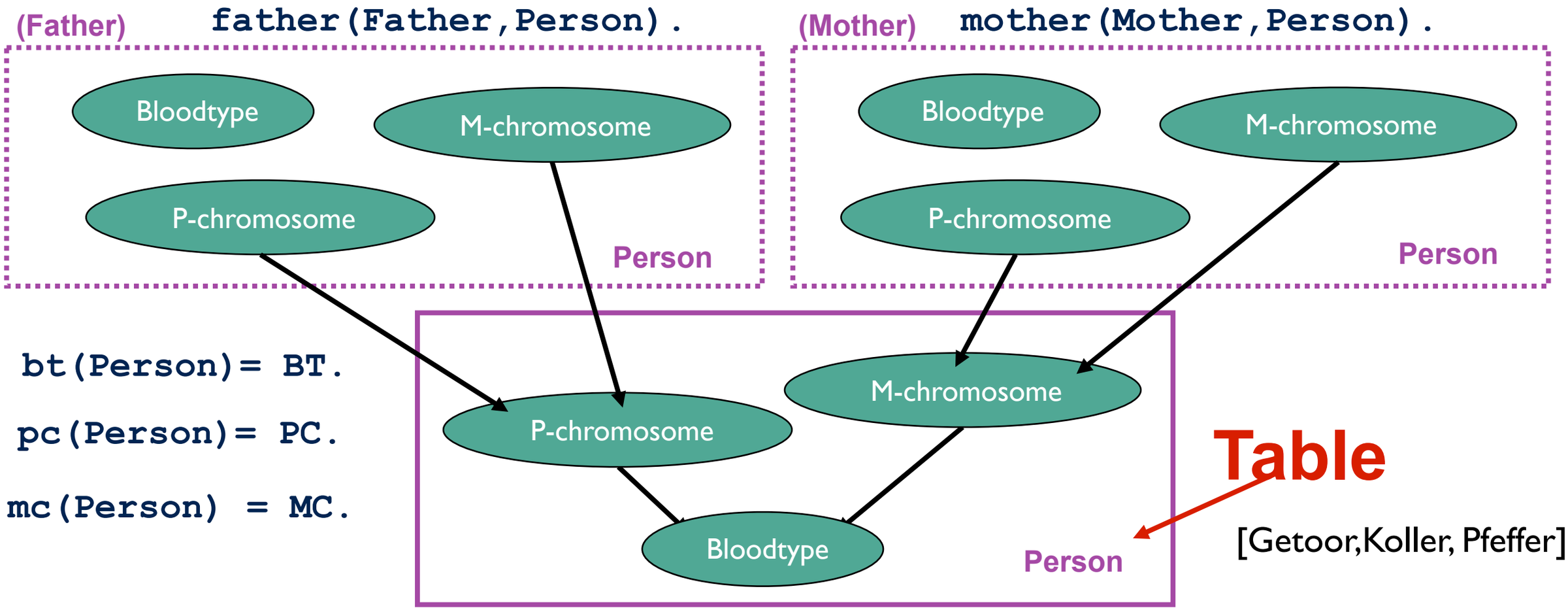
Dependencies (CPDs associated with):

```
bt(Person)=BT | pc(Person) =PC , mc(Person) =MC.
pc(Person) = PC | pc_father(Father)= PCf, mc_father(Father)= MCf.
```

View :

```
pc_father(Person) =PCf |   father(Father,Person),pc(Father)=PC.
                                        ...
```

Probabilistic Relational Models (PRMs)



[Getoor,Koller, Pfeffer]

Dependencies (CPDs associated with):

```
bt(Person)=BT | pc(Person) =PC , mc(Person) =MC.
pc(Person) = PC | pc_father(Father)= PCf, mc_father(Father)= MCf.
```

View :

```
pc_father(Person) =PCf |  father(Father,Person),pc(Father)=PC.
                                        ...
```

Probabilistic Relational Models (PRMs)
Bayesian Logic Programs (BLPs)

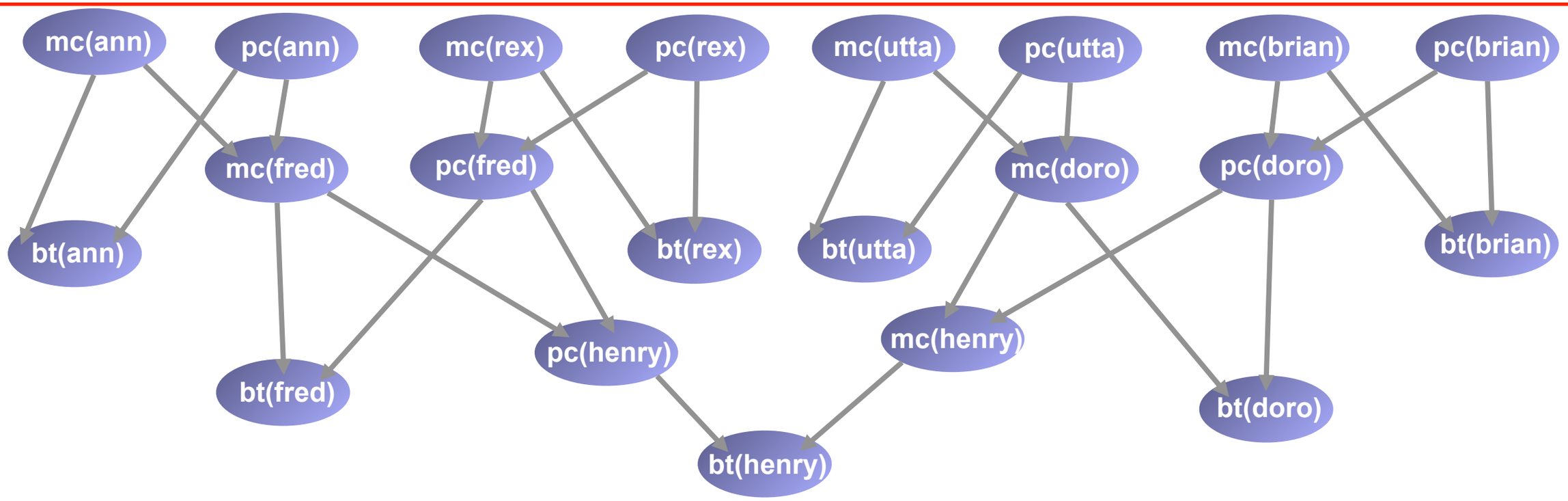
Extension

```
father(rex,fred).      mother(ann,fred).
father(brian,doro).    mother(utta, doro).
father(fred,henry).    mother(doro,henry).
```

Intension

```
pc_father(Person)= PCf |  father(Father,Person),pc(Father) = PC.
                                    ...
  mc(Person)=MC | pc_mother(Person)=PCm, pc_mother(Person)=MCm.
  pc(Person)=PC | pc_father(Person)=PCf, mc_father(Person)=MCf.
  bt(Person)=BT | pc(Person)=PC, mc(Person)=MC.
```

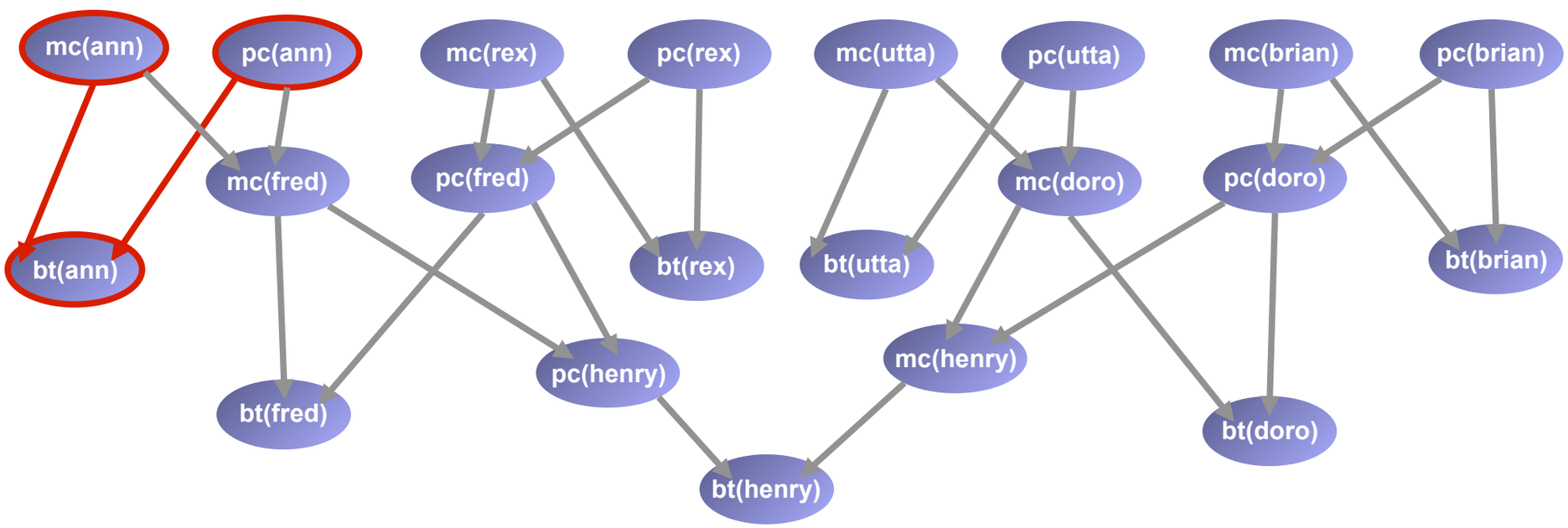
RV **State**



Answering Queries

P(`bt(ann)`**) ?**

Support Network

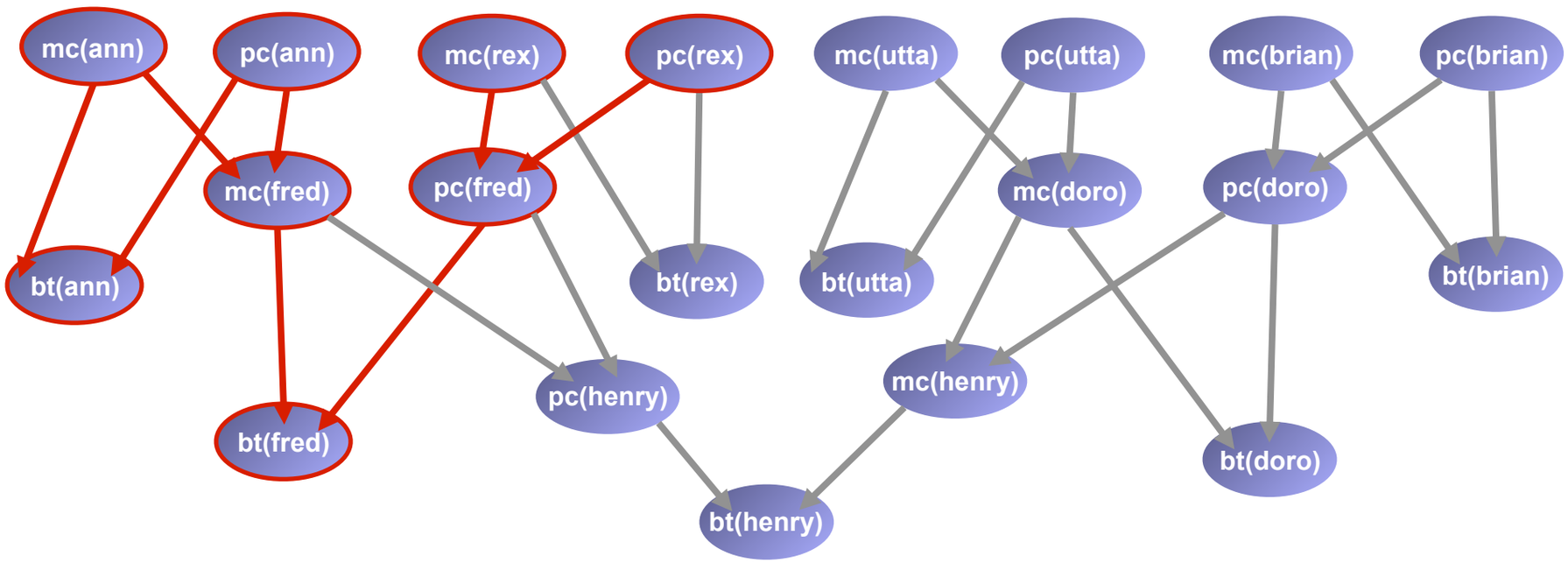


Answering Queries

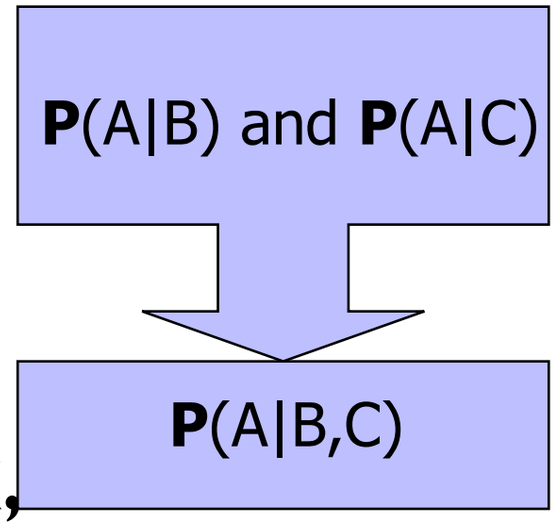
Bayes‘ rule

$$P(\texttt{bt(ann)} \mid \texttt{bt(fred)}) = \frac{P(\texttt{bt(ann)}, \texttt{bt(fred)})}{P(\texttt{bt(fred)})}$$

$P(\texttt{bt(ann)}, \texttt{bt(fred)})$?



Combining Rules



- Students reads two books
- Typical, noisy-or, noisy-max,

```
prepared(Student,Topic) | read(Student,Book),
                          discusses(Book,Topic).
```

Knowledge Based Model Construction

Extension + Intension =>Probabilistic Model

same intension used for multiple extensions

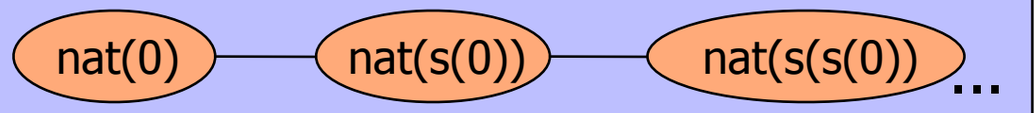
parameters are being shared / tied together

unification is essential

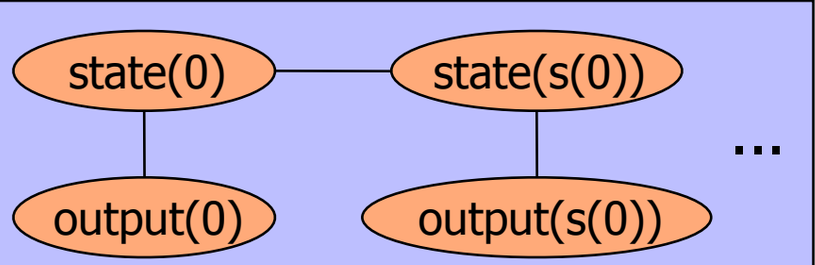
- learning becomes feasible
- max. likelihood parameter estimation & structure learning

Bayesian Logic Programs

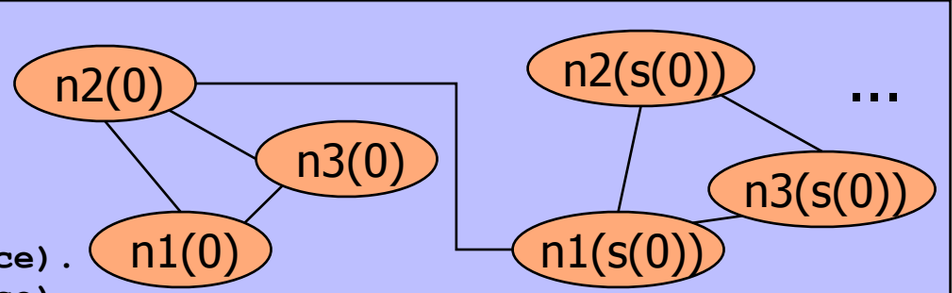
```
% apriori nodes
nat(0).

% aposteriori nodes
nat(s(X)) | nat(X).
```



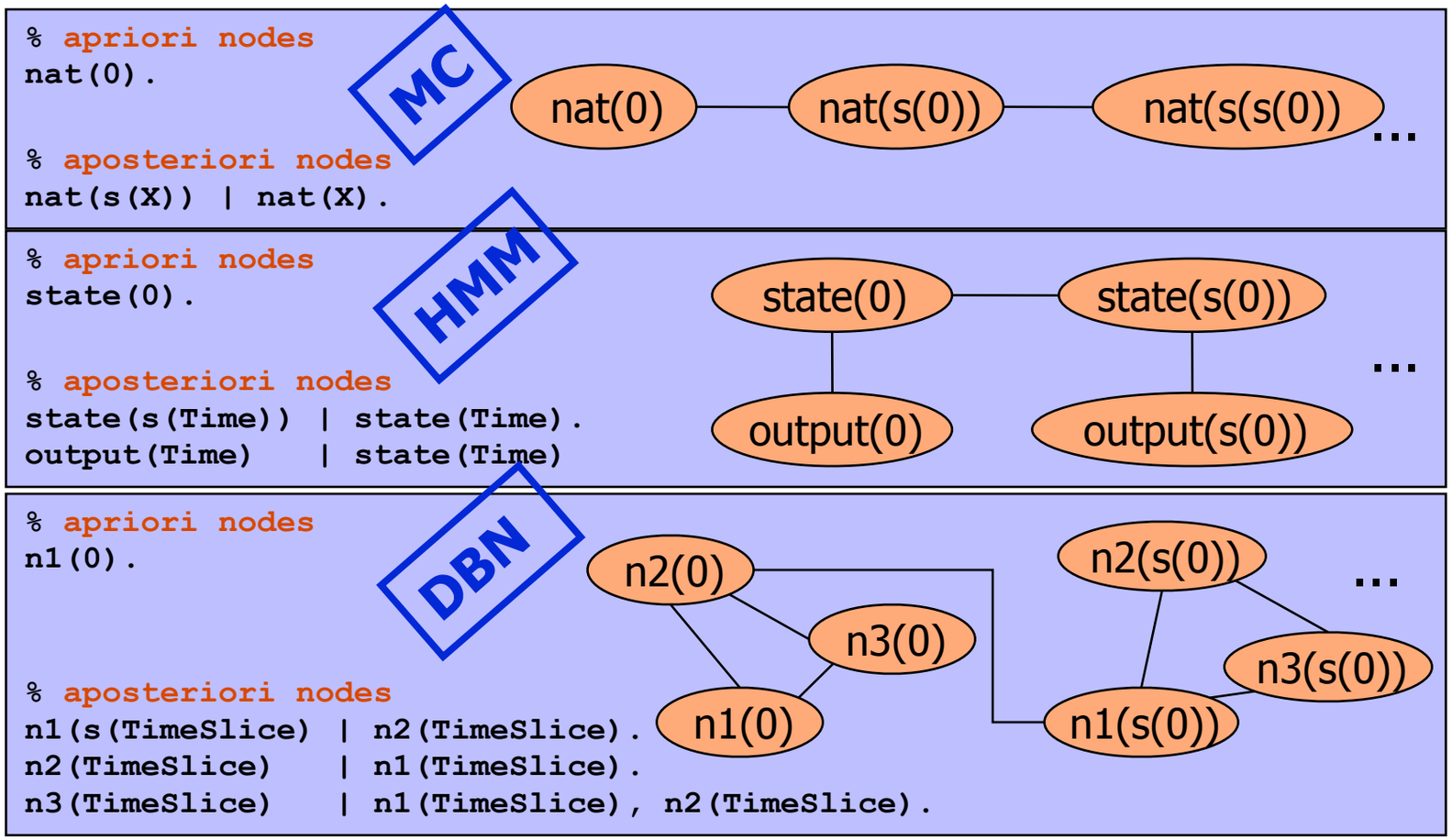

```
% apriori nodes
state(0).

% aposteriori nodes
state(s(Time)) | state(Time).
output(Time)   | state(Time)
```



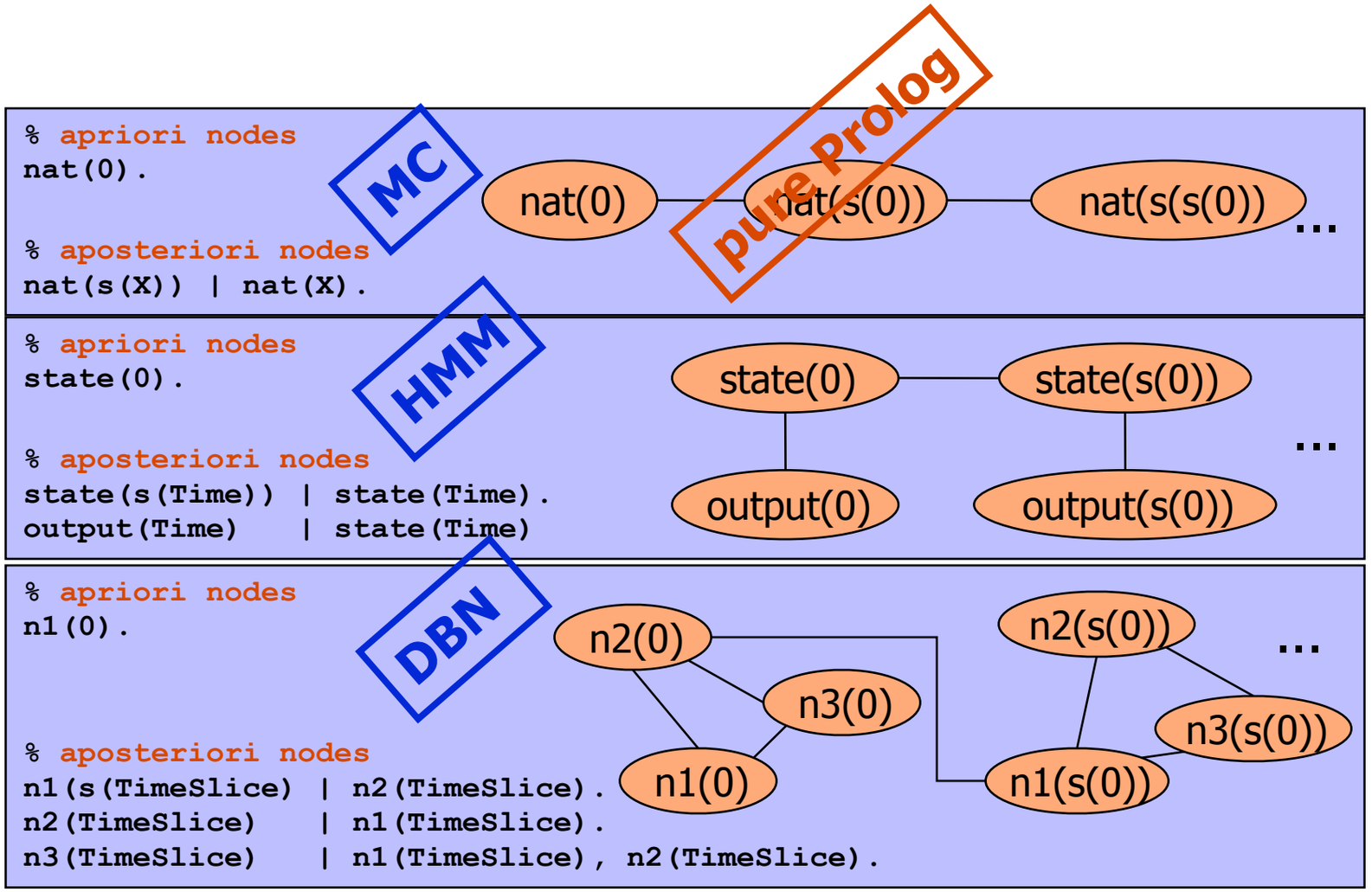

```
% apriori nodes
n1(0).

% aposteriori nodes
n1(s(TimeSlice) | n2(TimeSlice).
n2(TimeSlice)   | n1(TimeSlice).
n3(TimeSlice)   | n1(TimeSlice), n2(TimeSlice).
```



Bayesian Logic Programs



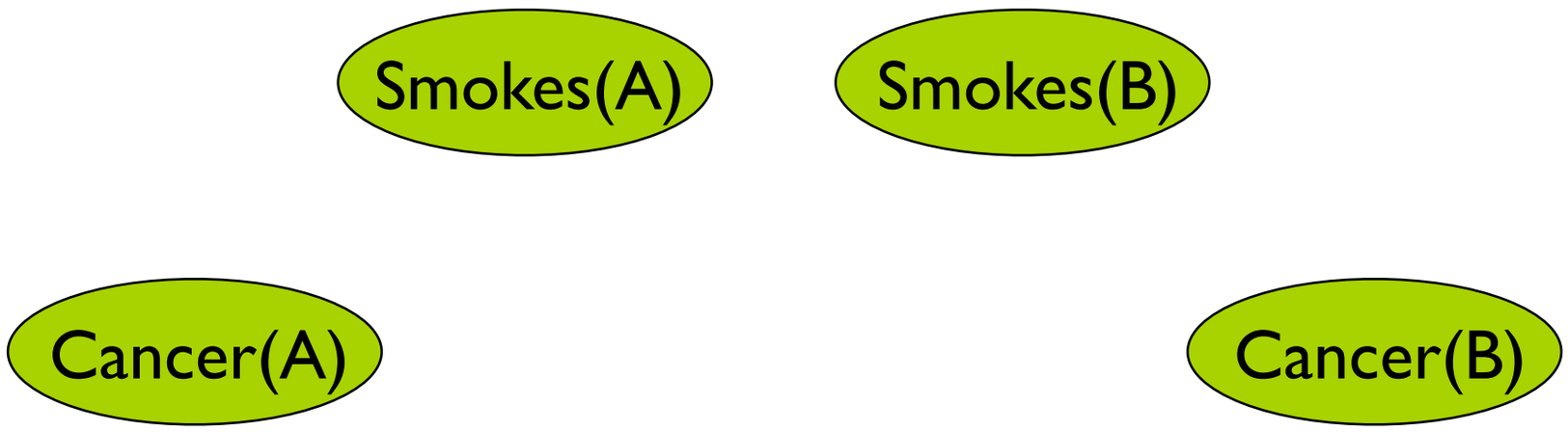
Bayesian Logic Programs



Markov Logic

1.5	$\forall x\ Smokes(x) \Rightarrow Cancer(x)$
1.1	$\forall x, y\ Friends(x, y) \Rightarrow (Smokes(x) \Leftrightarrow Smokes(y))$

Suppose we have two constants: **Anna** (A) and **Bob** (B)

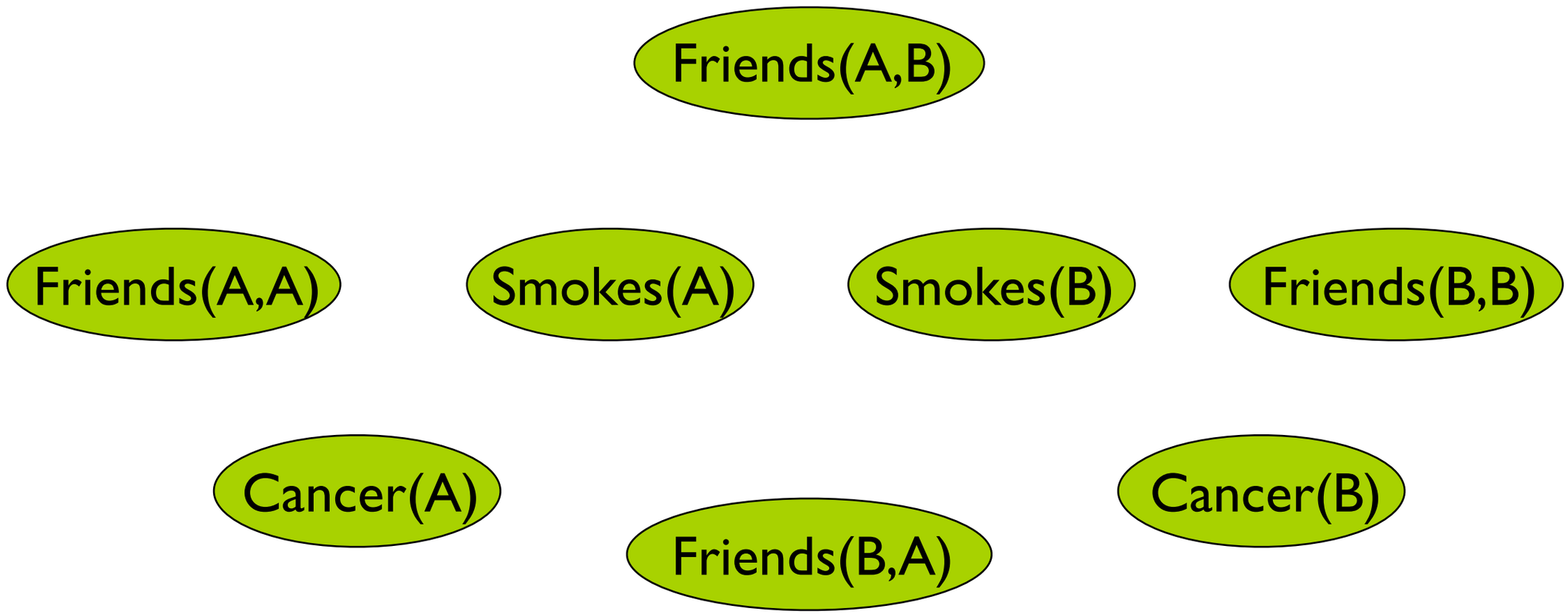


slides by Pedro Domingos

Markov Logic

1.5	$\forall x\ Smokes(x) \Rightarrow Cancer(x)$
1.1	$\forall x, y\ Friends(x, y) \Rightarrow (Smokes(x) \Leftrightarrow Smokes(y))$

Suppose we have two constants: **Anna** (A) and **Bob** (B)

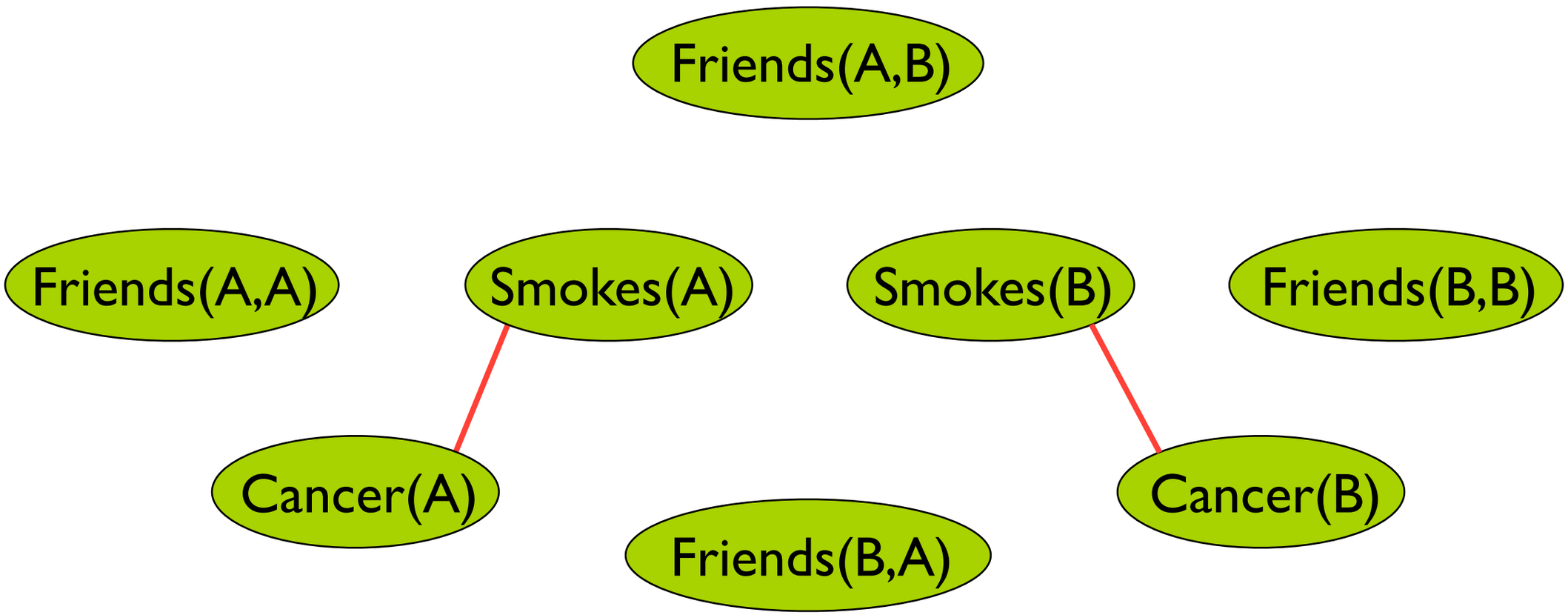


slides by Pedro Domingos

Markov Logic

1.5	$\forall x\ Smokes(x) \Rightarrow Cancer(x)$
1.1	$\forall x, y\ Friends(x, y) \Rightarrow (Smokes(x) \Leftrightarrow Smokes(y))$

Suppose we have two constants: **Anna** (A) and **Bob** (B)

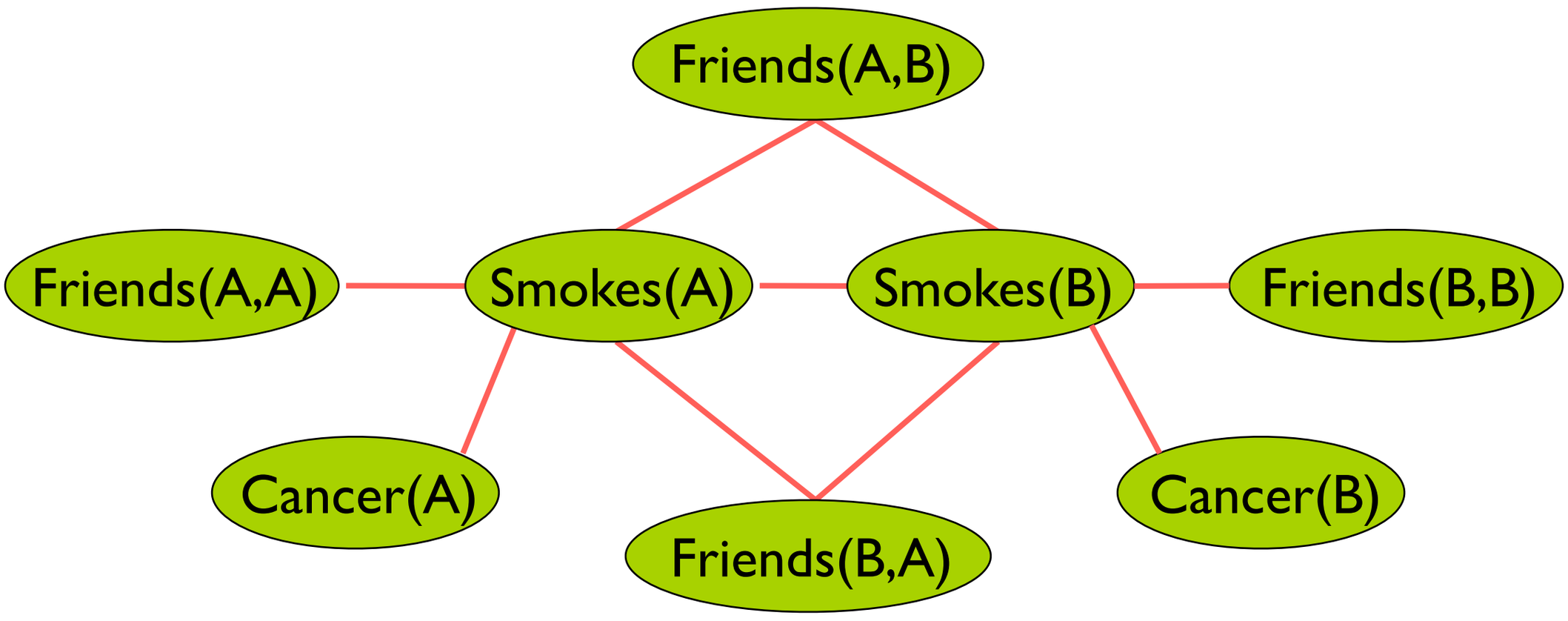


slides by Pedro Domingos

Markov Logic

1.5	$\forall x\ Smokes(x) \Rightarrow Cancer(x)$
1.1	$\forall x, y\ Friends(x, y) \Rightarrow (Smokes(x) \Leftrightarrow Smokes(y))$

Suppose we have two constants: **Anna** (A) and **Bob** (B)



slides by Pedro Domingos

Possible Worlds

A vocabulary

Smokes(Alice)	Smokes(Bob)	Friends(Alice,Bob)	Friends(Bob,Alice)
0	0	0	0
⋮	⋮	⋮	⋮
1	0	1	0
⋮	⋮	⋮	⋮
1	1	1	1

Possible worlds
Logical interpretations

Slides adapted from Guy Van den Broeck

Possible Worlds

A logical theory

$\forall x,y,\ \text{Smokes}(x) \wedge \text{Friends}(x,y) \Rightarrow \text{Smokes}(y)$

Smokes(Alice)	Smokes(Bob)	Friends(Alice,Bob)	Friends(Bob,Alice)	theory
0	0	0	0	1
⋮	⋮	⋮	⋮	⋮
1	0	1	0	0
⋮	⋮	⋮	⋮	⋮
1	1	1	1	1

Interpretations that satisfy the theory
Models

Slides adapted from Guy Van den Broeck

First-Order Model Counting

A logical theory

$\forall x,y,\ \text{Smokes}(x) \wedge \text{Friends}(x,y) \Rightarrow \text{Smokes}(y)$

Smokes(Alice)	Smokes(Bob)	Friends(Alice,Bob)	Friends(Bob,Alice)	theory
0	0	0	0	1
⋮	⋮	⋮	⋮	⋮
1	0	1	0	0
⋮	⋮	⋮	⋮	⋮
1	1	1	1	1

$\sum$

First-order model count
~#SAT

Slides Guy Van den Broeck

Markov Logic

- MLNs are a template for ground Markov Networks
- Probability of a world/interpretation
- If $n_i = 0$ then $P(x) = \frac{1}{Z}$

$$P(x) = \frac{1}{Z} \exp\left(\sum_i w_i n_i(x) \right)$$

Weight of formula *i*

No. of true groundings of formula *i* in *x*

Markov Logic

A Markov Logic theory

1.5 $\forall x,y$, Smokes(x) $\wedge$ Friends(x,y) $\Rightarrow$ Smokes(y)

Smokes(Alice)	Smokes(Bob)	Friends(Alice,Bob)	Friends(Bob,Alice)	theory
0	0	0	0	$\frac{1}{Z}exp(1.5*2)$
1	0	1	0	$\frac{1}{Z}exp(1.5*1)$
1	1	1	1	$\frac{1}{Z}exp(1.5*2)$

Slides adapted from Guy Van den Broeck

counting only substitutions for which X =/= Y
X=Alice, Y=Bob
X=Bob, Y=Alice

Markov Logic

A Markov Logic theory

1.5 $\forall x,y$, Smokes(x) $\wedge$ Friends(x,y) $\Rightarrow$ Smokes(y)

Smokes(Alice)	Smokes(Bob)	Friends(Alice,Bob)	Friends(Bob,Alice)	theory
0	0	0	0	$\frac{1}{Z}exp(1.5*2)$
1	0	1	0	$\frac{1}{Z}exp(1.5*1)$
1	1	1	1	$\frac{1}{Z}exp(1.5*2)$

$\sum$

Z
partition function

Slides adapted from Guy Van den Broeck

Weighted First-Order Model Counting

A logical theory and a weight function for predicates

Smokes(Alice)	Smokes(Bob)	Friends(Alice,Bob)	Friends(Bob,Alice)	theory	weight
0	0	0	0	1	$2 \cdot 2 \cdot 1 \cdot 1$
⋮	⋮	⋮	⋮	⋮	⋮
1	0	1	0	0	0
⋮	⋮	⋮	⋮	⋮	⋮
1	1	1	1	1	$1 \cdot 1 \cdot 4 \cdot 4$

$\sum$

Smokes → 1
¬Smokes → 2
Friends → 4
¬Friends → 1

Weighted first-order model count

Related to ProbLog Inference !

Parameter Learning

$$\frac{\partial}{\partial w_i} \log P_w(x) = n_i(x) - E_w\left[n_i(x)\right]$$

No. of times clause *i* is true in data

Expected no. times clause *i* is true according to MLN

Has been used for generative and discriminative learning
Many applications in networks, NLP, bioinformatics, …

PART VI

Lifted Inference

Lifted Inference

- exploiting symmetries & repeated structure
- reasoning on first order level as much as possible
- aiming at independence from number of objects
- approximation: grouping similar computations
- very active research area

Example:
First-Order Model Counting

1. Logical sentence

$$\text{Stress}(\text{Alice}) \Rightarrow \text{Smokes}(\text{Alice})$$

Domain

Alice

Slides Guy Van den Broeck

Example:
First-Order Model Counting

1. Logical sentence

Stress(Alice) ⇒ Smokes(Alice)

Domain

Alice

Stress(Alice)	Smokes(Alice)	Formula
0	0	1
0	1	1
1	0	0
1	1	1

Slides Guy Van den Broeck

Example:
First-Order Model Counting

1. Logical sentence

Stress(Alice) ⇒ Smokes(Alice)

Domain

Alice

→ 3 models

Example:
First-Order Model Counting

1. Logical sentence

$\text{Stress}(\text{Alice}) \Rightarrow \text{Smokes}(\text{Alice})$

Domain

Alice

→ 3 models

2. Logical sentence

$\forall x, \text{Stress}(x) \Rightarrow \text{Smokes}(x)$

Domain

Alice

Slides Guy Van den Broeck

Example:
First-Order Model Counting

1. Logical sentence

$$\text{Stress}(\text{Alice}) \Rightarrow \text{Smokes}(\text{Alice})$$

Domain

Alice

→ 3 models

2. Logical sentence

$$\forall x,\ \text{Stress}(x) \Rightarrow \text{Smokes}(x)$$

Domain

Alice

→ 3 models

Slides Guy Van den Broeck

Example:
First-Order Model Counting

2. Logical sentence

$\forall x,\ \text{Stress}(x) \Rightarrow \text{Smokes}(x)$

Domain

Alice

$\rightarrow$ 3 models

Example:
First-Order Model Counting

2. Logical sentence

$\forall x, \text{Stress}(x) \Rightarrow \text{Smokes}(x)$

Domain

Alice

$\rightarrow$ 3 models

3. Logical sentence

$\forall x, \text{Stress}(x) \Rightarrow \text{Smokes}(x)$

Domain

n people

Slides Guy Van den Broeck

Example:
First-Order Model Counting

2. Logical sentence

$\forall x, \text{Stress}(x) \Rightarrow \text{Smokes}(x)$

Domain

Alice

$\rightarrow$ 3 models

3. Logical sentence

$\forall x, \text{Stress}(x) \Rightarrow \text{Smokes}(x)$

Domain

n people

$\rightarrow 3^n$ models

Slides Guy Van den Broeck

Example:
First-Order Model Counting

3. Logical sentence

$\forall x, \text{Stress}(x) \Rightarrow \text{Smokes}(x)$

$\rightarrow 3^n$ models

Domain

n people

Slides Guy Van den Broeck

Example: First-Order Model Counting

3. Logical sentence

$\forall x, \text{Stress}(x) \Rightarrow \text{Smokes}(x)$

Domain

n people

$\rightarrow 3^n$ models

4. Logical sentence

$\forall y, \text{Parent}(y) \wedge \text{Female} \Rightarrow \text{Mother}(y)$

Domain

n people

Slides Guy Van den Broeck

Example: First-Order Model Counting

3. Logical sentence

$\forall x, \text{Stress}(x) \Rightarrow \text{Smokes}(x)$

Domain

n people

$\rightarrow 3^n$ models

4. Logical sentence

$\forall y, \text{Parent}(y) \wedge \text{Female} \Rightarrow \text{Mother}(y)$

Domain

n people

if Female: $\forall y, \text{Parent}(y) \Rightarrow \text{Mother}(y)$

Slides Guy Van den Broeck

Example: First-Order Model Counting

3. Logical sentence

$\forall x, \text{Stress}(x) \Rightarrow \text{Smokes}(x)$

Domain

n people

$\rightarrow 3^n$ models

4. Logical sentence

$\forall y, \text{Parent}(y) \wedge \text{Female} \Rightarrow \text{Mother}(y)$

Domain

n people

if not Female:

True

Slides Guy Van den Broeck

Example: First-Order Model Counting

3. Logical sentence

$\forall x, \text{Stress}(x) \Rightarrow \text{Smokes}(x)$

Domain

n people

$\rightarrow 3^n$ models

4. Logical sentence

$\forall y, \text{Parent}(y) \wedge \text{Female} \Rightarrow \text{Mother}(y)$

Domain

n people

$\rightarrow (3^n+4^n)$ models

Slides Guy Van den Broeck

Example:
First-Order Model Counting

4. Logical sentence

$\forall y,\ \text{Parent}(y) \wedge \text{Female} \Rightarrow \text{Mother}(y)$

Domain

n people

$\rightarrow (3^n+4^n)$ models

Example: First-Order Model Counting

4. Logical sentence

$\forall y,\ Parent(y) \wedge Female \Rightarrow Mother(y)$

Domain

n people

$\rightarrow (3^n+4^n)$ models

5. Logical sentence

$\forall x,y,\ ParentOf(x,y) \wedge Female(x) \Rightarrow MotherOf(x,y)$

Domain

n people

Slides Guy Van den Broeck

Example: First-Order Model Counting

4. Logical sentence

$\forall y,\ \text{Parent}(y) \wedge \text{Female} \Rightarrow \text{Mother}(y)$

Domain

n people

$\rightarrow (3^n+4^n)$ models

5. Logical sentence

$\forall x,y,\ \text{ParentOf}(x,y) \wedge \text{Female}(x) \Rightarrow \text{MotherOf}(x,y)$

Domain

n people

$\rightarrow (3^n+4^n)^n$ models

Slides Guy Van den Broeck

Example:
First-Order Model Counting

6. Logical sentence

$\forall x,y,\ \text{Smokes}(x) \wedge \text{Friends}(x,y) \Rightarrow \text{Smokes}(y)$

Domain

n people

Slides Guy Van den Broeck

Example:
First-Order Model Counting

6. Logical sentence

$\forall x,y,\ \text{Smokes}(x) \wedge \text{Friends}(x,y) \Rightarrow \text{Smokes}(y)$

Domain

n people

Slides Guy Van den Broeck

Example:
First-Order Model Counting

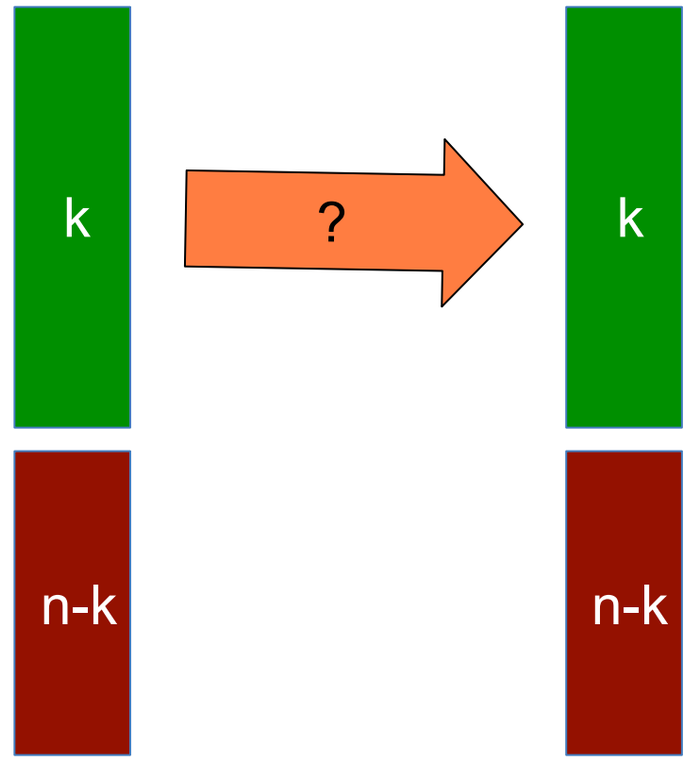
6. Logical sentence

$\forall x,y,\ \text{Smokes}(x) \wedge \text{Friends}(x,y) \Rightarrow \text{Smokes}(y)$

Domain

n people

Smokes Alice
Smokes Bob
Smokes Charlie
Smokes Dave
Smokes Eve



Slides Guy Van den Broeck

Example:
First-Order Model Counting

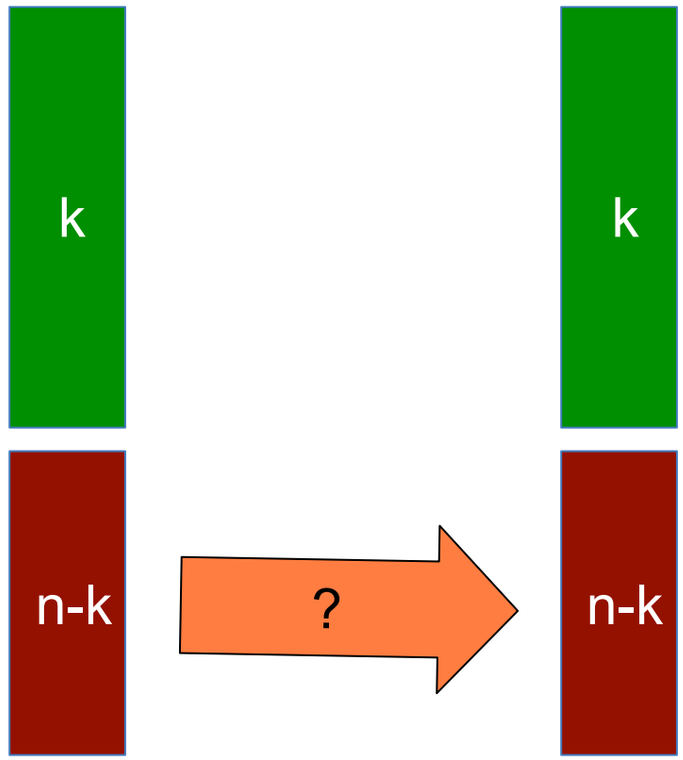
6. Logical sentence

$\forall x,y,\ \text{Smokes}(x) \wedge \text{Friends}(x,y) \Rightarrow \text{Smokes}(y)$

Domain

n people

Smokes Alice
Smokes Bob
Smokes Charlie
Smokes Dave
Smokes Eve



Slides Guy Van den Broeck

Example:
First-Order Model Counting

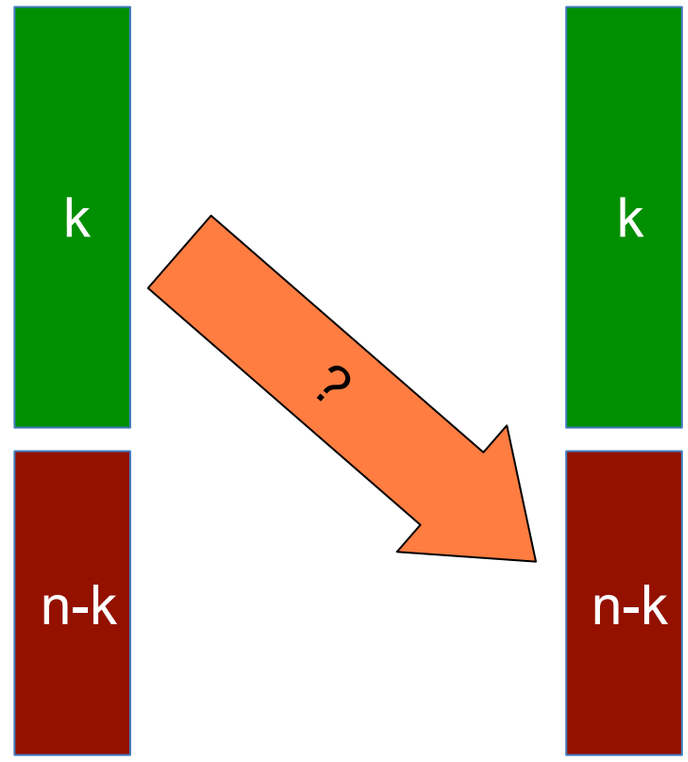
6. Logical sentence

$\forall x,y,\ \text{Smokes}(x) \wedge \text{Friends}(x,y) \Rightarrow \text{Smokes}(y)$

Domain

n people

Smokes Alice
Smokes Bob
Smokes Charlie
Smokes Dave
Smokes Eve



Slides Guy Van den Broeck

Example:
First-Order Model Counting

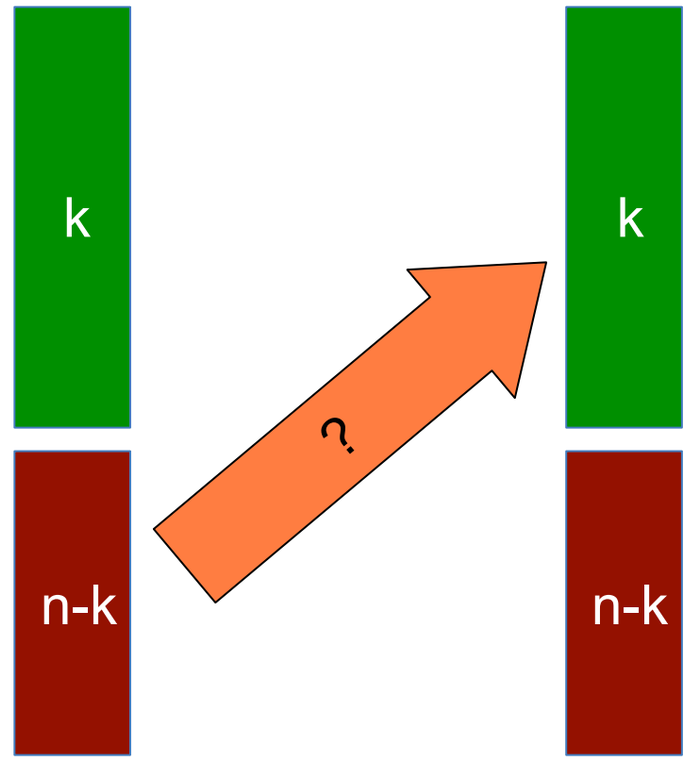
6. Logical sentence

$\forall x,y,\ \text{Smokes}(x) \wedge \text{Friends}(x,y) \Rightarrow \text{Smokes}(y)$

Domain

n people

Smokes Alice
Smokes Bob
Smokes Charlie
Smokes Dave
Smokes Eve



Slides Guy Van den Broeck

Example:
First-Order Model Counting

6. Logical sentence

$\forall x,y,\ \text{Smokes}(x) \wedge \text{Friends}(x,y) \Rightarrow \text{Smokes}(y)$

Domain

n people

$\rightarrow \quad 2^{n^2 - k(n-k)}$ models

Slides Guy Van den Broeck

Example:
First-Order Model Counting

6. Logical sentence

$\forall x,y,\ \mathrm{Smokes}(x) \wedge \mathrm{Friends}(x,y) \Rightarrow \mathrm{Smokes}(y)$

Domain

n people

$\rightarrow \quad 2^{n^2 - k(n-k)}$ models

Slides Guy Van den Broeck

Example:
First-Order Model Counting

6. Logical sentence

$\forall x,y,\ \text{Smokes}(x) \wedge \text{Friends}(x,y) \Rightarrow \text{Smokes}(y)$

Domain

n people

$\rightarrow \quad 2^{n^2 - k(n-k)}$ models

$\rightarrow \quad \binom{n}{k} 2^{n^2 - k(n-k)}$ models

Slides Guy Van den Broeck

Example:
First-Order Model Counting

6. Logical sentence

$\forall x,y,\ \text{Smokes}(x) \wedge \text{Friends}(x,y) \Rightarrow \text{Smokes}(y)$

Domain

n people

$\rightarrow \quad 2^{n^2-k(n-k)}$ models

$\rightarrow \quad \binom{n}{k} 2^{n^2-k(n-k)}$ models

Slides Guy Van den Broeck

Example:
First-Order Model Counting

6. Logical sentence

$\forall x,y,\ \text{Smokes}(x) \wedge \text{Friends}(x,y) \Rightarrow \text{Smokes}(y)$

Domain

n people

$\rightarrow \quad 2^{n^2-k(n-k)}$ models

$\rightarrow \quad \binom{n}{k} 2^{n^2-k(n-k)}$ models

In total $\sum_{k=0}^{n} \binom{n}{k} 2^{n^2-k(n-k)}$

Slides Guy Van den Broeck

The Full Pipeline

MLN

3.14 $\text{Smokes}(x) \wedge \text{Friends}(x,y) \Rightarrow \text{Smokes}(y)$

Slides Guy Van den Broeck

The Full Pipeline

MLN

3.14 $\quad \text{Smokes}(x) \wedge \text{Friends}(x,y) \Rightarrow \text{Smokes}(y)$

$\forall x,y,\ F(x,y) \Leftrightarrow [\ \text{Smokes}(x) \wedge \text{Friends}(x,y) \Rightarrow \text{Smokes}(y)\ ]$

Relational Logic

The Full Pipeline

MLN

3.14 $\quad \text{Smokes}(x) \wedge \text{Friends}(x,y) \Rightarrow \text{Smokes}(y)$

$\forall x,y,\ F(x,y) \Leftrightarrow [\ \text{Smokes}(x) \wedge \text{Friends}(x,y) \Rightarrow \text{Smokes}(y)\ ]$

Relational Logic

$\text{Smokes} \rightarrow 1$
$\neg\text{Smokes} \rightarrow 1$
$\text{Friends} \rightarrow 1$
$\neg\text{Friends} \rightarrow 1$
$F \rightarrow \exp(3.14)$
$\neg F \rightarrow 1$

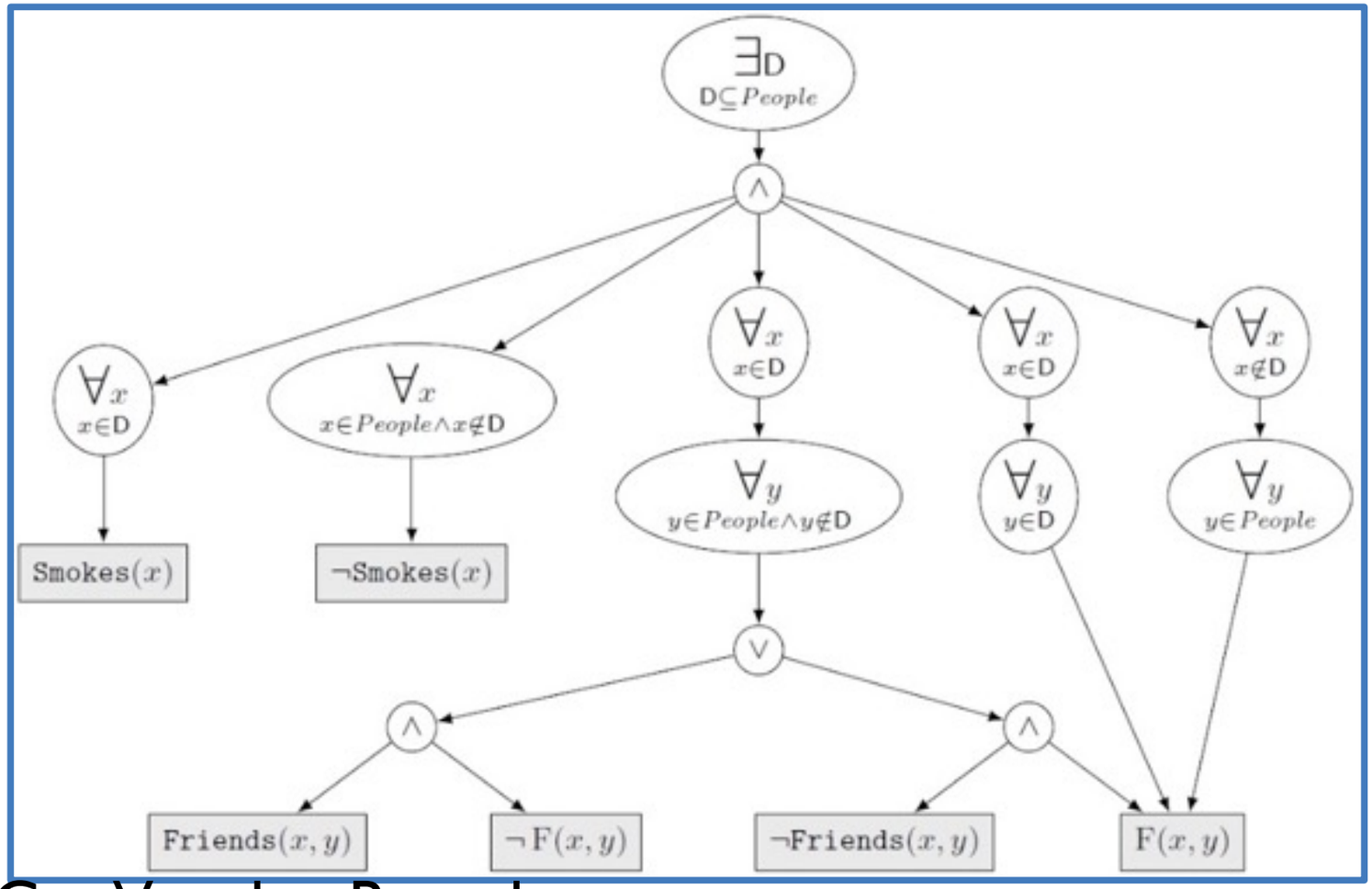
Weight Function

Slides Guy Van den Broeck

The Full Pipeline

$$\forall x,y,\ F(x,y) \Leftrightarrow [\ Smokes(x) \wedge Friends(x,y) \Rightarrow Smokes(y)\ ]$$

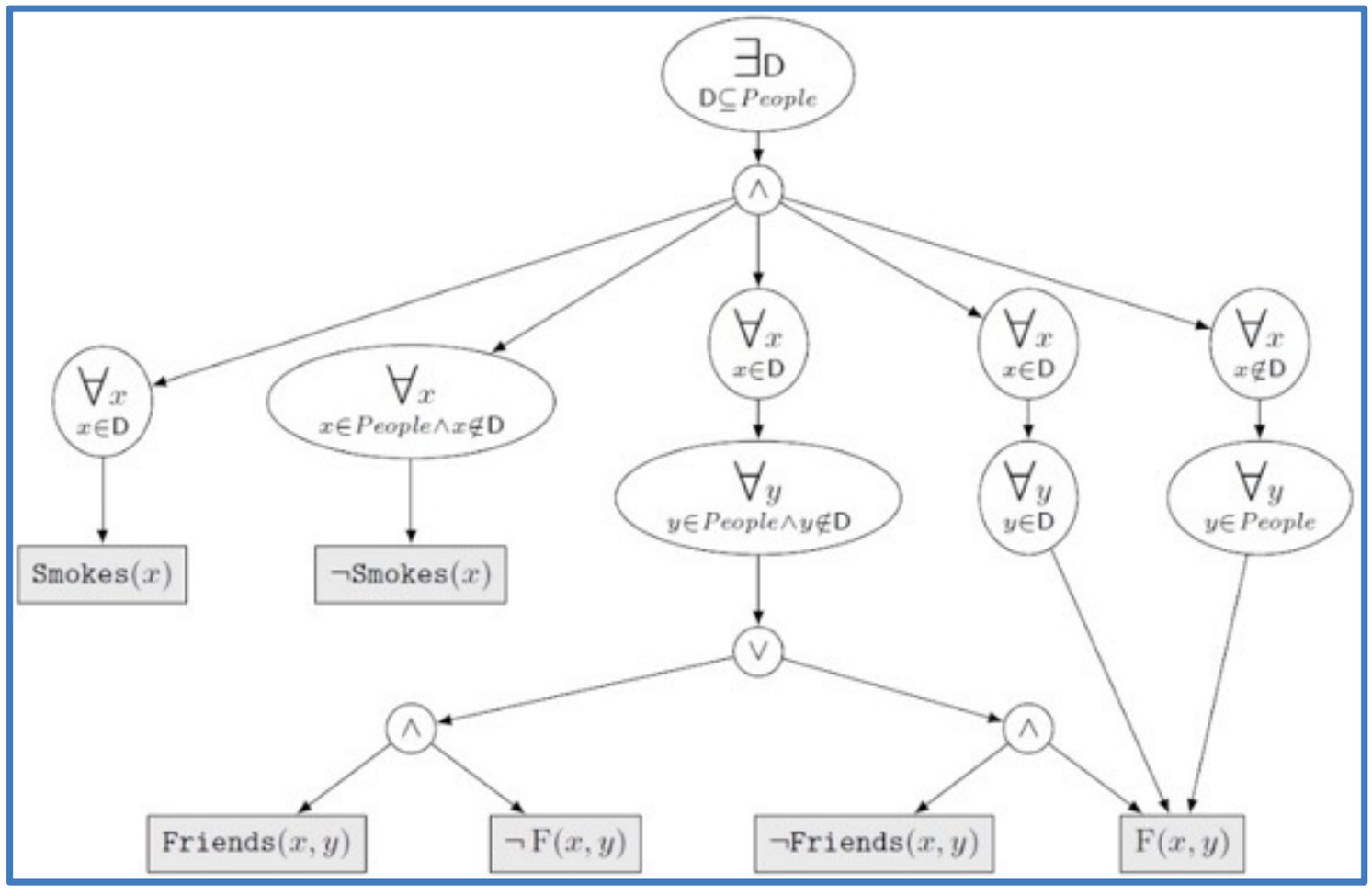
Relational Logic



First-Order
d-DNNF Circuit

Slides Guy Van den Broeck

The Full Pipeline



First-Order d-DNNF Circuit

Smokes → 1
¬Smokes → 1
Friends → 1
¬Friends → 1
F → exp(3.14)
¬F → 1

Weight Function

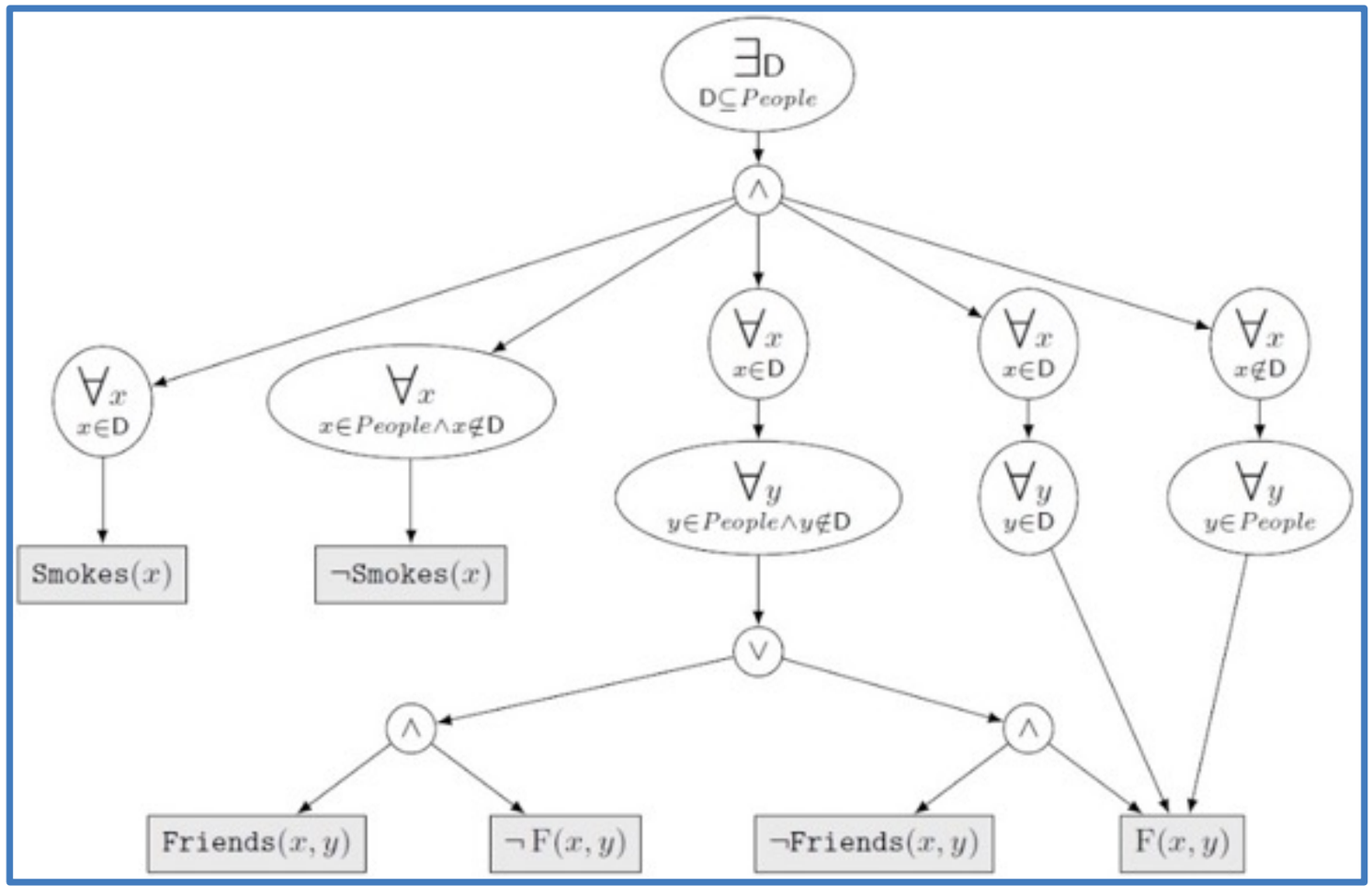
Alice
Bob
Charlie

Domain

Weighted First-Order Model Count is 1479.85

Slides Guy Van den Broeck

The Full Pipeline



First-Order d-DNNF Circuit

Smokes → 1
¬Smokes → 1
Friends → 1
¬Friends → 1
F → exp(3.14)
¬F → 1

Weight Function

Alice
Bob
Charlie

Domain

Weighted First-Order Model Count is 1479.85

Circuit evaluation is polynomial in domain size!

Slides Guy Van den Broeck

Advanced Topics

- parameter estimation
- lifted graphical models and KBMC
- complexity results from prob. db
- lifted inference

Roadmap

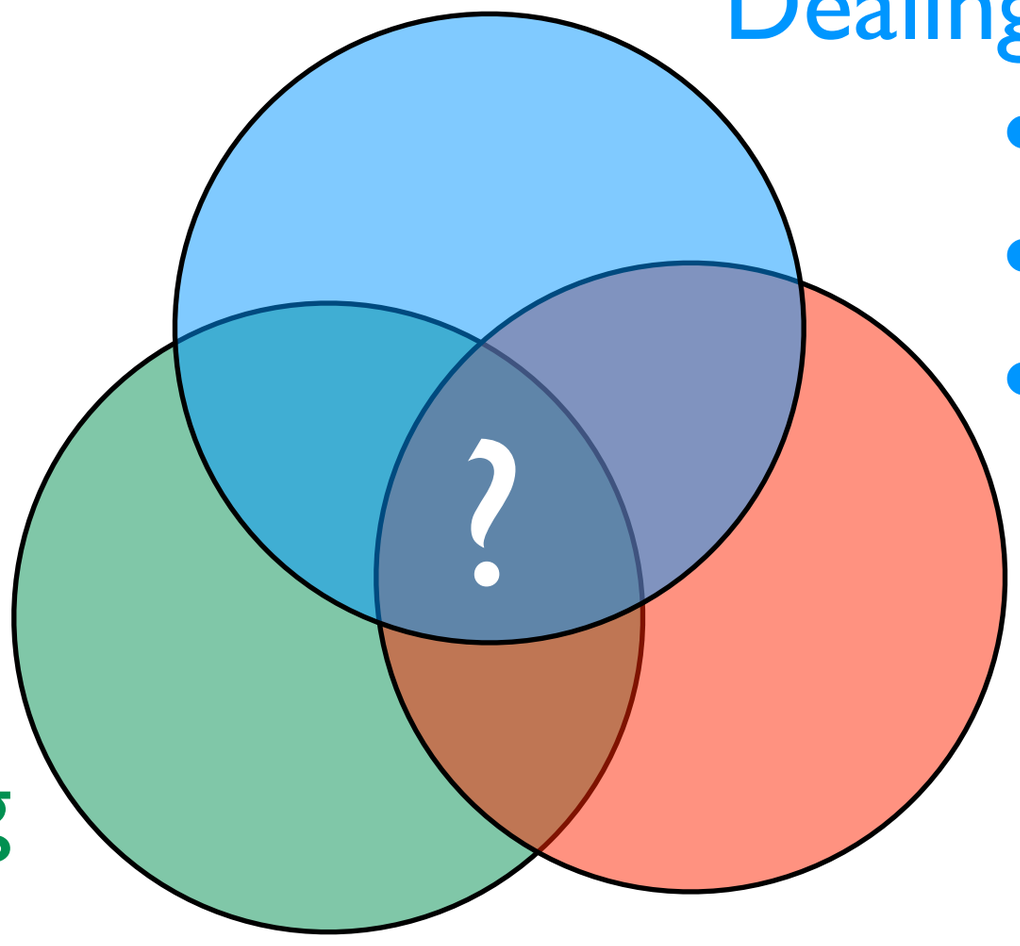
- Modeling
- Inference
- Learning & Dynamics
- Advanced Inference and KBMC

... with some detours on the way

A key question in AI:

Reasoning with relational data

- logic
- databases
- programming
- ...



Dealing with uncertainty

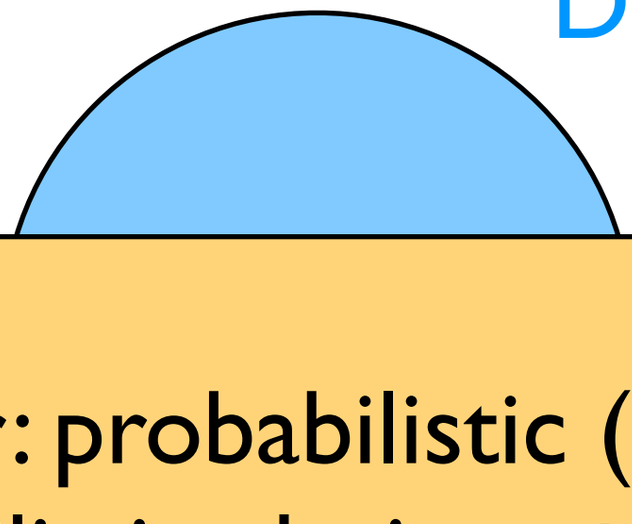
- probability theory
- graphical models
- ...

Learning

- parameters
- structure

Statistical relational learning, probabilistic logic learning, probabilistic programming, ...

A key question in AI:



Dealing with uncertainty
- probability theory

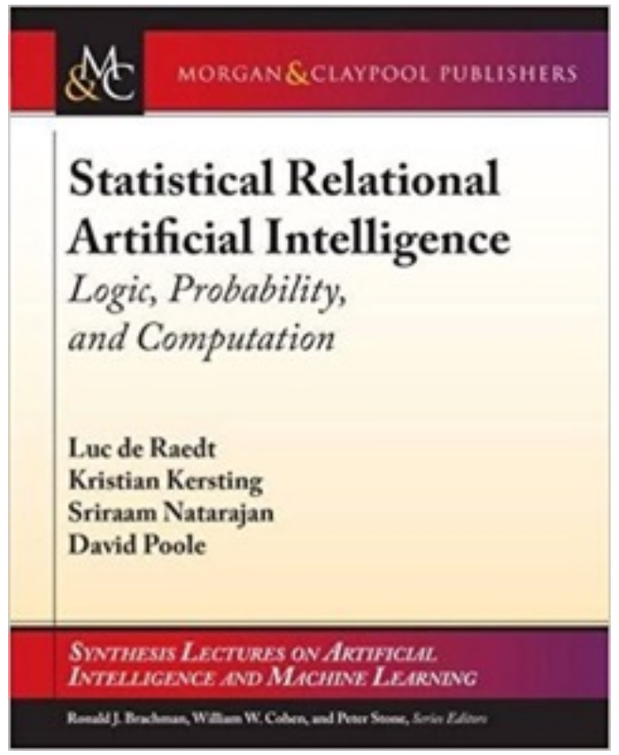
Reas
relat
- log
- da
- pr
- ...

- Our answer: probabilistic (logic) programming
  = probabilistic choices + (logic) program
- Many languages, systems, applications, ...
- ... and much more to do!

structure

Statistical relational learning, probabilistic logic learning, probabilistic programming, ...

Further Reading

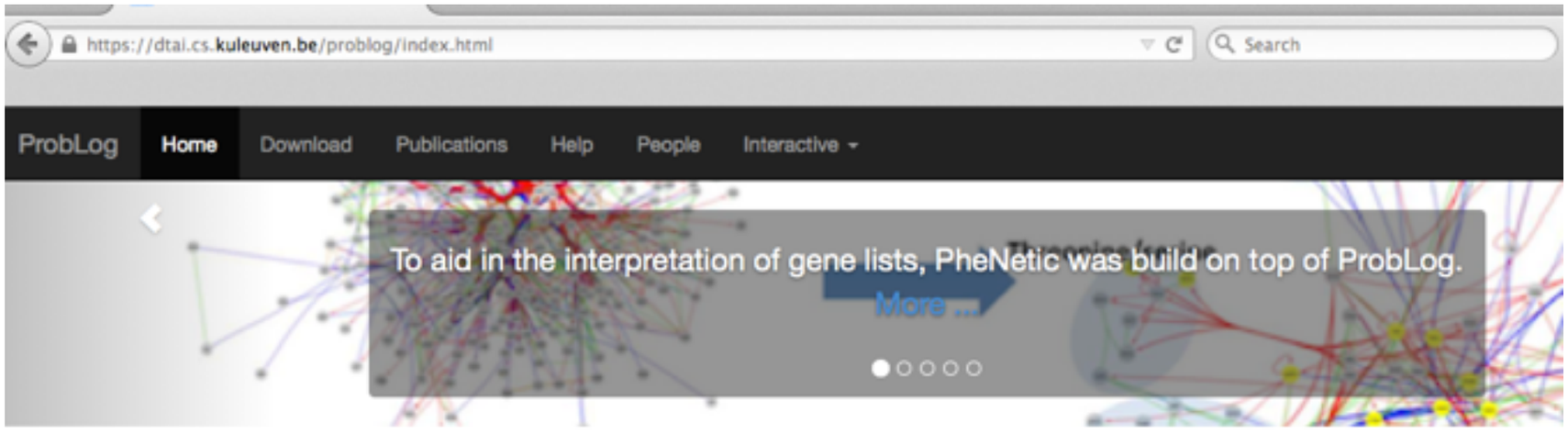


- One book
- Three websites to start
  - http://probmods.org/ Probabilistic Models of Cognition — Church
  - http://dtai.cs.kuleuven.be/problog/ — check also [DR & Kimmig, MLJ 15]
  - http://alchemy.cs.washington.edu/ —Markov Logic, check also [Domingos & Lowd] Markov Logic, Morgan Claypool.

Maurice Bruynooghe
Bart Demoen
Anton Dries
Daan Fierens
Jason Filippou
Bernd Gutmann
Manfred Jaeger
Gerda Janssens
Kristian Kersting
Angelika Kimmig
Theofrastos Mantadelis
Wannes Meert
Bogdan Moldovan
Siegfried Nijssen
Davide Nitti
Joris Renkens
Kate Revoredo
Ricardo Rocha
Vitor Santos Costa
Dimitar Shterionov
Ingo Thon
Hannu Toivonen
Guy Van den Broeck
Mathias Verbeke
Jonas Vlasselaer

Thanks !

http://dtai.cs.kuleuven.be/problog



PLP Systems

- **PRISM** http://sato-www.cs.titech.ac.jp/prism/
- **ProbLog2** http://dtai.cs.kuleuven.be/problog/
- Yap Prolog http://www.dcc.fc.up.pt/~vsc/Yap/ includes
  - **ProbLog1**
  - **cplint** https://sites.google.com/a/unife.it/ml/cplint
  - **CLP(BN)**
  - **LP2**
- **PITA** in XSB Prolog http://xsb.sourceforge.net/
- **AILog2** http://artint.info/code/ailog/ailog2.html
- **SLPs** http://stoics.org.uk/~nicos/sware/pepl
- **contdist** http://www.cs.sunysb.edu/~cram/contdist/
- **DC** https://code.google.com/p/distributional-clauses
- **WFOMC** http://dtai.cs.kuleuven.be/ml/systems/wfomc

References

Bach SH, Broecheler M, Getoor L, O'Leary DP (2012) Scaling MPE inference for constrained continuous Markov random fields with consensus optimization. In: Proceedings of the 26th Annual Conference on Neural Information Processing Systems (NIPS-12)

Broecheler M, Mihalkova L, Getoor L (2010) Probabilistic similarity logic. In: Proceedings of the 26th Conference on Uncertainty in Artificial Intelligence (UAI-10)

Bryant RE (1986) Graph-based algorithms for Boolean function manipulation. IEEE Transactions on Computers 35(8):677–691

Cohen SB, Simmons RJ, Smith NA (2008) Dynamic programming algorithms as products of weighted logic programs. In: Proceedings of the 24th International Conference on Logic Programming (ICLP-08)

Cussens J (2001) Parameter estimation in stochastic logic programs. Machine Learning 44(3):245–271

De Maeyer D, Renkens J, Cloots L, De Raedt L, Marchal K (2013) Phenetic: network-based interpretation of unstructured gene lists in e. coli. Molecular BioSystems 9(7):1594–1603

De Raedt L, Kimmig A (2013) Probabilistic programming concepts. CoRR abs/1312.4328

De Raedt L, Kimmig A, Toivonen H (2007) ProbLog: A probabilistic Prolog and its application in link discovery. In: Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI-07)

De Raedt L, Frasconi P, Kersting K, Muggleton S (eds) (2008) Probabilistic Inductive Logic Programming — Theory and Applications, Lecture Notes in Artificial Intelligence, vol 4911. Springer

Eisner J, Goldlust E, Smith N (2005) Compiling Comp Ling: Weighted dynamic programming and the Dyna language. In: Proceedings of the Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP-05)

Fierens D, Blockeel H, Bruynooghe M, Ramon J (2005) Logical Bayesian networks and their relation to other probabilistic logical models. In: Proceedings of the 15th International Conference on Inductive Logic Programming (ILP-05)

Fierens D, Van den Broeck G, Bruynooghe M, De Raedt L (2012) Constraints for probabilistic logic programming. In: Proceedings of the NIPS Probabilistic Programming Workshop

Fierens D, Van den Broeck G, Renkens J, Shterionov D, Gutmann B, Thon I, Janssens G, De Raedt L (2014) Inference and learning in probabilistic logic programs using weighted Boolean formulas. Theory and Practice of Logic Programming (TPLP) FirstView

Getoor L, Friedman N, Koller D, Pfeffer A, Taskar B (2007) Probabilistic relational models. In: Getoor L, Taskar B (eds) An Introduction to Statistical Relational Learning, MIT Press, pp 129–174

Goodman N, Mansinghka VK, Roy DM, Bonawitz K, Tenenbaum JB (2008) Church: a language for generative models. In: Proceedings of the 24th Conference on Uncertainty in Artificial Intelligence (UAI-08)

Gutmann B, Thon I, De Raedt L (2011a) Learning the parameters of probabilistic logic programs from interpretations. In: Proceedings of the 22nd European Conference on Machine Learning (ECML-11)

Gutmann B, Thon I, Kimmig A, Bruynooghe M, De Raedt L (2011b) The magic of logical inference in probabilistic programming. Theory and Practice of Logic Programming (TPLP) 11((4–5)):663–680

Huang B, Kimmig A, Getoor L, Golbeck J (2013) A flexible framework for probabilistic models of social trust. In: Proceedings of the International Conference on Social Computing, Behavioral-Cultural Modeling, and Prediction (SBP-13)

Jaeger M (2002) Relational Bayesian networks: A survey. Linköping Electronic Articles in Computer and Information Science 7(015)

Kersting K, Raedt LD (2001) Bayesian logic programs. CoRR cs.AI/0111058

Kimmig A, Van den Broeck G, De Raedt L (2011a) An algebraic Prolog for reasoning about possible worlds. In: Proceedings of the 25th AAAI Conference on Artificial Intelligence (AAAI-11)

Kimmig A, Demoen B, De Raedt L, Santos Costa V, Rocha R (2011b) On the implementation of the probabilistic logic programming language ProbLog. Theory and Practice of Logic Programming (TPLP) 11:235–262

Koller D, Pfeffer A (1998) Probabilistic frame-based systems. In: Proceedings of the 15th National Conference on Artificial Intelligence (AAAI-98)

McCallum A, Schultz K, Singh S (2009) FACTORIE: Probabilistic programming via imperatively defined factor graphs. In: Proceedings of the 23rd Annual Conference on Neural Information Processing Systems (NIPS-09)

Milch B, Marthi B, Russell SJ, Sontag D, Ong DL, Kolobov A (2005) Blog: Probabilistic models with unknown objects. In: Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI-05)

Moldovan B, De Raedt L (2014) Occluded object search by relational affordances. In: IEEE International Conference on Robotics and Automation (ICRA-14)

Moldovan B, Moreno P, van Otterlo M, Santos-Victor J, De Raedt L (2012) Learning relational affordance models for robots in multi-object manipulation tasks. In: IEEE International Conference on Robotics and Automation (ICRA-12)

Muggleton S (1996) Stochastic logic programs. In: De Raedt L (ed) Advances in Inductive Logic Programming, IOS Press, pp 254–264

Nitti D, De Laet T, De Raedt L (2013) A particle filter for hybrid relational domains. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS-13)

Nitti D, De Laet T, De Raedt L (2014) Relational object tracking and learning. In: IEEE International Conference on Robotics and Automation (ICRA), June 2014

Pfeffer A (2001) IBAL: A probabilistic rational programming language. In: Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI-01)

Pfeffer A (2009) Figaro: An object-oriented probabilistic programming language. Tech. rep., Charles River Analytics

Poole D (2003) First-order probabilistic inference. In: Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI-03)

Richardson M, Domingos P (2006) Markov logic networks. Machine Learning 62(1-2):107–136

Santos Costa V, Page D, Cussens J (2008) CLP(*BN*): Constraint logic programming for probabilistic knowledge. In: De Raedt et al (2008), pp 156–188

Sato T (1995) A statistical learning method for logic programs with distribution semantics. In: Proceedings of the 12th International Conference on Logic Programming (ICLP-95)

Sato T, Kameya Y (2001) Parameter learning of logic programs for symbolic-statistical modeling. J Artif Intell Res (JAIR) 15:391–454

Sato T, Kameya Y (2008) New advances in logic-based probabilistic modeling by prism. In: Probabilistic Inductive Logic Programming, pp 118–155

Skarlatidis A, Artikis A, Filiopou J, Paliouras G (2014) A probabilistic logic programming event calculus. Theory and Practice of Logic Programming (TPLP) FirstView

Suciu D, Olteanu D, Ré C, Koch C (2011) Probabilistic Databases. Synthesis Lectures on Data Management, Morgan & Claypool Publishers

Taskar B, Abbeel P, Koller D (2002) Discriminative probabilistic models for relational data. In: Proceedings of the 18th Conference on Uncertainty in Artificial Intelligence (UAI-02)

Thon I, Landwehr N, De Raedt L (2008) A simple model for sequences of relational state descriptions. In: Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases (ECML/PKDD-08)

Thon I, Landwehr N, De Raedt L (2011) Stochastic relational processes: Efficient inference and applications. Machine Learning 82(2):239–272

Van den Broeck G, Thon I, van Otterlo M, De Raedt L (2010) DTProbLog: A decision-theoretic probabilistic Prolog. In: Proceedings of the 24th AAAI Conference on Artificial Intelligence (AAAI-10)

Van den Broeck G, Taghipour N, Meert W, Davis J, De Raedt L (2011) Lifted probabilistic inference by first-order knowledge compilation. In: Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI-11)

Vennekens J, Verbaeten S, Bruynooghe M (2004) Logic programs with annotated disjunctions. In: Proceedings of the 20th International Conference on Logic Programming (ICLP-04)

Vennekens J, Denecker M, Bruynooghe M (2009) CP-logic: A language of causal probabilistic events and its relation to logic programming. Theory and Practice of Logic Programming (TPLP) 9(3):245–308

Wang WY, Mazaitis K, Cohen WW (2013) Programming with personalized pagerank: a locally groundable first-order probabilistic logic. In: Proceedings of the 22nd ACM International Conference on Information and Knowledge Management (CIKM-13)